Abstract

A dimension in a Data Warehouse (DW) is a set of elements connected by a hierarchical relationship. The elements are used to view summaries of data at different levels of abstraction. In order to support an efficient processing of such summaries, a dimension is usually required to satisfy different classes of integrity constraints. In scenarios where the constraints properly capture the semantics of the DW data, but they are not satisfied by the dimension, it arises the problem of repairing (correcting) the dimension. In this paper, we study the problem of repairing a dimension in the context of two main classes of integrity constraints: strictness and covering constraints. We introduce the notion of minimal repair of a dimension: a new dimension that is consistent with respect to the set of integrity constraints, which is obtained by applying a minimal number of updates to the original dimension. We study the complexity of obtaining minimal repairs, and show how they can be characterized using Datalog programs with weak constraints under the stable model semantics.

Key words: data warehouses, dimensions, integrity constraints, inconsistency, Datalog programs, stable models

1. Introduction

Data Warehouses (DWs) are data repositories that integrate data from different sources, and keep historical data for analysis and decision support [19]. When generating reports, it is of central importance for these systems to compute summaries of data in a simple and efficient way. In order to do this, DWs organize data according to the multidimensional model. In the multidimensional model, dimensions reflect the perspectives upon which facts are viewed. Facts correspond to events which are usually associated to numeric values known as measures, and are referenced using the dimension elements. Dimensions are modeled as hierarchies of elements, where each element belongs to a category. The categories are also organized into a hierarchy called hierarchy schema.

Example 1. Consider a company that manages an online repository of research articles. The company maintains a DW to generate summaries used to analyze the download behavior of its
users. The DW contains the dimensions Time and Publication. The Time dimension is structured using a hierarchy schema with a bottom category Date, which goes to the category Month, which in turn goes to the category Year. On the other hand, the Publication dimension is structured using the hierarchy schema shown in Figure 1(a), where the Article category goes to Journal, which in turn goes to Subject ACM, and then to Area. The article repository also has an internal classification for articles, so that Journal is connected to Subject Int, which in turn goes to Area. The top category is All. Figure 1(b) shows the elements of the Publication dimension, along with the relations between elements of different categories, called rollup relations. For example, TODS (Transaction of Database Systems) and DKE (Data Knowledge Engineering) are elements of category Journal and DB (DataBases), DM (Data Management Systems) and DT (Database Theory) are elements of category Subject Int. For each edge in the hierarchy schema, there is a rollup relation. For example, the rollup relation from the category Journal to the category Subject ACM contains the following pairs or elements: (TODS, H2) and (DKE, G2).

The fact table of the DW is shown in Figure 1(c). Each fact stored in the table represents the number of times an article was downloaded in a given date. As an example, article A1 was downloaded three times on January 1, 2007. The hierarchical structure of dimensions allows users to access facts at different levels of granularity. As an example, using the aforementioned DW it is easy to compute summaries such as: number of times that each article was downloaded per month, or number of downloads broken down by area and year.

1.1. Problem Statement

In order to compute summaries efficiently, DWs use pre-computed summaries at low level categories to derive summaries at higher level categories. Two main classes of integrity constraints, strictness and covering constraints, are used to check whether such computations, called summarizations, are correct [57, 41, 34]. In a consistent summarization, each fact is aggregated once and not more than once.

Strictness constraints are used to require rollup relations to be functions. If a rollup relation between categories A and B is required to be strict, then there cannot exist an element in A connected to two different elements in B. As an example, in the Publication dimension (Figure 1(a) and (b)) the rollup relation from Journal to Subject ACM is strict. In contrast, the rollup relation from Journal to Subject Int is not strict, because TODS is connected to two different elements in the category Subject Int. Covering constraints are used to require a rollup relation from category A to B to connect all the elements in A to at least one element in B. As an example,
Example 2. Figure 2 shows four summaries extracted from the DW of Figure 1. Each summary $S_C$ represents the total number of downloads broken down by some category $C$. The summary $S_{Area}$ can be correctly derived from the summary $S_{Article}$ by summing up the number of downloads for each group of articles that go to the same area in the dimension (in this case $A_1$ goes to IS and $A_2$ goes to MC). The correctness of this derivation follows from the fact that the rollup relation from $Article$ to $Area$ is strict and covering. Similarly, $S_{Area}$ can be correctly derived from $S_{Subject ACM}$, and the correctness of this derivation follows from the fact that the rollup relation from $Article$ to $Subject ACM$ and the rollup relation from $Subject ACM$ to $Area$ are both strict and covering. However, $S_{Area}$ cannot be correctly derived from $S_{Subject Int}$ since such derivation would yield 0 downloads for the element MC, while the correct number is 7. This derivation is incorrect because the rollup relation from $Article$ to $Subject Int$ is not covering.

It has been stated by many researchers and practitioners that the dimensions that arise in real-world applications do not always have strict and covering rollup relations. As an example, we may have products in a product dimension, that belong to different elements in a type category, or may have documents in a document dimension that belong to different topics in a topic category. On the other hand, some products may not be associated to a type, or some documents may not belong to any topic. The flexibility to represent rollup relations has been incorporated as a central feature of several dimension models [34, 52, 55, 56, 45]. However, in order to keep the ability to find consistent summarizations, it is important for DW systems to know the set of strictness and covering constraints that hold, which can be achieved by allowing DW designers to formulate explicitly the constraints when the dimensions are modeled.

Example 3. Consider the dimension of Figure 3. Suppose that the DW administrator considers that the constraint $Journal \rightarrow Area$, which states that the rollup relation from $Journal$ to $Area$ is strict, should be satisfied by the dimension. However, the dimension does not satisfy this constraint. Indeed, the element DKE goes to two different elements: IS and MC, which are both in the category $Area$. The DW administrator considers that the constraint properly captures the semantic of the data, and therefore its violation indicates the presence of errors. Therefore, the
dimension should be repaired (corrected) in order to resolve the inconsistency. There are several possible repairs that could be computed and presented to the administrator as possible solutions to the problem. As an example, the dimension can be repaired by deleting (DKE, DT) and adding (DKE, DM) to the rollup relation from Journal to Subject_Int. Another option is to delete (DKE, H2) and add (DKE, G2) to the rollup relation from Journal to Subject_ACM (Figure 4 shows different repairs for this dimension). The problem that arises is to compute repairs obtained by applying a minimum amount of changes to the original dimension.

1.2. Contributions

In this paper, we study the problem of restoring consistency of a dimension that does not satisfy a set of strictness and covering constraints. We introduce the notion of a minimal repair of a dimension which does not satisfy a set of constraints. A minimal repair is defined as a new dimension that satisfies the constraints and which is obtained by applying a minimal number of changes to the original dimension. The contributions include the following:

- Formalization of dimension (minimal) repairs with respect to covering and strictness constraints.
- Complexity analysis of the problem of computing a minimal repair, and show that in general the problem is NP-hard but that there is a special case in which it can be done in polynomial time. It is also shown that repairs always exist.
- Provide a logic programming specification to compute repairs of general dimensions based on Datalog programs with stable model semantics with weak constraints [42]. These programs solve an NP-complete problem, therefore they will be in general inefficient. However, they provide a starting point from which other, probably approximate algorithms, can be compared for quality and efficiency.
- Suggest alternative repair semantics that take into consideration user preferences such as avoiding the modification of some roll-up relations, assigning different weights to changes in different relations, etc.

This work is based in an extended abstract appeared in the Proceedings of the IV Alberto Mendelzon Workshop on Foundations of Data Management [13].

1.3. Outline

The rest of the paper is organized as follows: Section 2 presents a formalization of DW dimensions and strictness and covering constraints. Next, Section 3 presents the notion of repair, variations of it and analyzes relevant complexity problems. Section 4 presents the logic programs
to compute repairs. Finally sections 5 and 6 present related work and the conclusions of the paper. The proofs of all the results can be found in the appendix.

2. Preliminaries

In Section 2.1 we formalize dimensions using standard concepts from multidimensional models obtained from [16, 36, 38, 55], with some minor modifications to facilitate presentation. The notation presented is general enough to represent dimensions compatible with the requirements to model complex multidimensional data introduced by [55], including the modeling of non-strict, and non-covering (or heterogeneous) dimensions. In addition and we formalize strictness and covering constraints (Section 2.2).

2.1. Dimensions

We start by defining the notion of hierarchy schema.

**Definition 1.** [Hierarchy Schema] A hierarchy schema $\mathcal{H}$ consists of a pair $(C_\mathcal{H}, \triangleright_\mathcal{H})$, where $(C_\mathcal{H}, \triangleright_\mathcal{H})$ is an acyclic directed graph. Vertices in the set $C_\mathcal{H}$ are categories and the edges $\triangleright_\mathcal{H}$ represent the child/parent relations between categories. The transitive and reflexive closure of $\triangleright_\mathcal{H}$ is denoted by $\triangleright_\mathcal{H}^\ast$. The set of categories $C_\mathcal{H}$ contains a distinguished top category denoted $\text{All}_\mathcal{H}$, which is reachable from every other category in $C_\mathcal{H}$ and has no outgoing edges, that is, there is no category $c_i \in C_\mathcal{H}$ such that $(\text{All}_\mathcal{H}, c_i) \in \triangleright_\mathcal{H}$ and for every $c_j \in C_\mathcal{H}$, $(c_j, \text{All}_\mathcal{H}) \in \triangleright_\mathcal{H}^\ast$. Sometimes, we will write $c_a \triangleright_\mathcal{H} c_b$ instead of $(c_a, c_b) \in \triangleright_\mathcal{H}$.

In real-world DWs categories usually have attributes [21]. As an example, for the Journal category in the dimension of Figure 1 we may consider attributes such as editorial, editor, and so on. Nevertheless, for simplification purposes, we assume that categories do not have attributes, which is a common assumption when modeling dimensions. The distinguished and unique top category is introduced in several models [30, 38, 16, 36, 52, 53].

**Example 4.** The hierarchy schema $\mathcal{H} = (C_\mathcal{H}, \triangleright_\mathcal{H})$, depicted in Figure 1(a), is as follows:

$C_\mathcal{H} = \{\text{Article}, \text{Journal}, \text{Subject}_\text{Int}, \text{Subject}_\text{ACM}, \text{Area}, \text{All}\}; \text{All}_\mathcal{H} = \text{All};$ and

$\triangleright_\mathcal{H} = \{(\text{Article}, \text{Journal}), (\text{Journal}, \text{Subject}_\text{Int}), (\text{Journal}, \text{Subject}_\text{ACM}), (\text{Subject}_\text{Int}, \text{Area}), (\text{Subject}_\text{ACM}, \text{Area}), (\text{Area}, \text{All})\}$. 

**Definition 2.** [Dimension] A dimension $\mathcal{D}$ is a tuple $(\mathcal{H}_\mathcal{D}, \mathcal{E}_\mathcal{D}, \text{Cat}_\mathcal{D}, <_\mathcal{D})$, where $\mathcal{H}_\mathcal{D} = (C_{\mathcal{H}_\mathcal{D}}, \triangleright_{\mathcal{H}_\mathcal{D}})$ is a hierarchy schema; $\mathcal{E}_\mathcal{D}$ is a set of constants, called elements; $\text{Cat}_\mathcal{D} : \mathcal{E}_\mathcal{D} \rightarrow C_{\mathcal{H}_\mathcal{D}}$ is a function that defines to which category each element in $\mathcal{E}_\mathcal{D}$ belongs; and the relation $<_{\mathcal{D}} \subseteq \mathcal{E}_\mathcal{D} \times \mathcal{E}_\mathcal{D}$ represents the child/parent relations between elements of different categories. We denote by $<_\mathcal{D}$ the reflexive and transitive closure of $<_{\mathcal{D}}$. The following conditions hold: (i) $\text{All}_{\mathcal{H}_\mathcal{D}}$ is the only element in category $\text{All}_{\mathcal{H}_\mathcal{D}}$, (ii) for all pair of elements $a, b \in \mathcal{E}_\mathcal{D}$ if $a <_{\mathcal{D}} b$ then $\text{Cat}_\mathcal{D}(a) \triangleright_{\mathcal{H}_\mathcal{D}} \text{Cat}_\mathcal{D}(b)$.

Condition (ii) ensures that the child/parent relation ($<_\mathcal{D}$) only connects elements of categories that are connected in the schema.

**Example 5.** Let $\mathcal{D} = (\mathcal{H}_\mathcal{D}, \mathcal{E}_\mathcal{D}, \text{Cat}_\mathcal{D}, <_{\mathcal{D}})$ be the dimension given in Figure 1. Then $\mathcal{H}_\mathcal{D}$ is as defined in Example 4 and:
\[ R_i \] and \[ R_e \] \[ Strict and Covering Rollup Relations \]

Let \[ \text{Definition 4.} \] Let \( R \in \mathcal{R} \) be a rollup relation, then:

(i) \( R \) is strict if \( R(c_i, c_j) \) is a function, i.e., if for all elements \( x, y, z \in \mathcal{E}_D \), if \( (x, y) \in R(c_i, c_j) \) and \( (x, z) \in R(c_i, c_j) \) then \( y = z \).

(ii) \( R \) is covering if for all elements \( e \in \mathcal{E}_D \) such that \( \text{Cat}(e) = c_i \), there exists an element \( e' \in \mathcal{E}_D \) such that \( \text{Cat}(e') = c_j \) and \( (e, e') \in R(c_i, c_j) \).

A dimension is strict if all its rollup relations are strict. Otherwise, the dimension is said to be non-strict. Similarly, we use the notions of covering and non-covering dimensions. Covering dimensions are also called homogeneous and non-covering dimensions are called heterogeneous dimensions.

The dimension model considered in this paper satisfies the desiderata for modeling complex dimensions proposed in \([52, 53, 55]\): (i) it captures explicitly the hierarchies in the dimensions; (ii) it allows multiple hierarchies of categories (i.e., a category may have more than one parent category in the hierarchy schema); (iii) it supports aggregation semantics (this can be done by verifying consistency of summarizations using the integrity constraints we will present in the next section or after applying transformations over dimension as proposed in \([53]\)); (iv) it supports non-strict rollup relations; (v) it supports non-onto rollup relations (i.e., some elements in a category may not be connected to elements in categories below); (vi) it allows the representation of non-covering rollup relations.

2.2. Integrity Constraints

The vast majority of research and industrial applications of DWs consider dimensions that are strict and covering \([19, 40, 21]\). For strict and covering dimensions \([34]\) summarization operations that aggregate facts between two categories that are connected in the hierarchy schema are always correct. However, it has been shown that in some real situations dimensions fail to satisfy these conditions \([53, 55, 35, 34, 46, 33]\). In these cases, in order to keep the ability to verify summarizability, it is useful to allow the DW administrator to specify integrity constraints to identify rollup relations that are strict or covering \([35]\) (covering is also known as roll-up completeness in the context of summarizability \([48]\)).
Definition 5. [Strictness Constraints, Covering Constraints] Let $\mathcal{H} = (C_{\mathcal{H}}, \rightarrow_{\mathcal{H}})$ be a hierarchy schema and let $\mathcal{D} = (\mathcal{H}_D, E_D, \text{Cat}_D, <_D)$ be a dimension such that $\mathcal{H}_D = \mathcal{H}$.

(i) A strictness constraint over $\mathcal{H}$ is an expression of the form $c_i \rightarrow c_j$ where $c_i, c_j \in C_{\mathcal{H}}$ and $c_i \rightarrow^*_{\mathcal{H}} c_j$. The dimension $\mathcal{D}$ satisfies the strictness constraint $c_i \rightarrow c_j$ if and only if the rollup relation $R_D(c_i, c_j)$ is strict.

(ii) A covering constraint over $\mathcal{H}$ is an expression of the form $c_i \Rightarrow c_j$ where $c_i, c_j \in C_{\mathcal{H}}$ and $c_i \rightarrow^*_{\mathcal{H}} c_j$. The dimension $\mathcal{D}$ satisfies the covering constraint $c_i \Rightarrow c_j$ if and only if the rollup relation $R_D(c_i, c_j)$ is covering.

We denote by $\Sigma_\rightarrow(\mathcal{H})$ and $\Sigma_\Rightarrow(\mathcal{H})$ the set of all possible strictness constraints and covering constraints, respectively, over hierarchy schema $\mathcal{H}$. We say that a dimension $\mathcal{D}$ satisfies a set of constraints $\Sigma$ if $\mathcal{D}$ satisfies every constraint in $\Sigma$. Otherwise, we say that the dimension $\mathcal{D}$ is inconsistent with respect to $\Sigma$.

Example 7. Let $\mathcal{D}$ be the publication dimension depicted in Figure 1(b). This dimension satisfies the following set $\Sigma$ of strict constraints: $\{\text{Article} \rightarrow \text{Journal}, \text{Article} \rightarrow \text{Area}, \text{Journal} \rightarrow \text{Area}, \text{Subject}\_\text{Int} \rightarrow \text{Area}, \text{Subject}\_\text{ACM} \rightarrow \text{Area}\}$. However, dimension $\mathcal{D}$ does not satisfy the constraint $\text{Journal} \rightarrow \text{Subject}\_\text{Int}$, since an element of $\text{Journal}$ may reach more than one element in $\text{Subject}\_\text{Int}$. In contrast, dimension given in Figure 3 is inconsistent with respect to $\Sigma$ since it violates the strict constraints $\text{Article} \rightarrow \text{Area}$ and $\text{Journal} \rightarrow \text{Area}$. Indeed, the rollup relations $R_D(\text{Article, Area})$ and $R_D(\text{Journal, Area})$ are non-strict:

<table>
<thead>
<tr>
<th>$R_D(\text{Article, Area})$</th>
<th>$R_D(\text{Journal, Area})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$ IS</td>
<td>TODS IS</td>
</tr>
<tr>
<td>$A_2$ IS</td>
<td>DKE IS</td>
</tr>
<tr>
<td>$A_3$ MC</td>
<td>DKE MC</td>
</tr>
</tbody>
</table>

The problem of determining if a dimension $\mathcal{D}$ satisfies a set of constraints can be solved in polynomial time. In fact, a simple algorithm could compute $<_D$, and then for each pair of categories $c_i, c_j$ such that $c_i \rightarrow^*_D c_j$ it could project $<^*_D$ to obtain the rollup relation from $c_i, c_j$, and finally check if the resulting relation is strict or covering. Known algorithms that compute the transitive closure of digraphs, which run in $O(|E_D|^2 + |E_D| \times |<_D|)$ (e.g., [25, 50]), can be used to obtain $<_D$.

3. Dimension Repairs

If a dimension $\mathcal{D}$ does not satisfy a set of constraints $\Sigma$, we would like to compute possible repairs of $\mathcal{D}$, that is, dimensions over the hierarchy schema of $\mathcal{D}$ that satisfy $\Sigma$.

Example 8. Let $\mathcal{D}$ be the dimension given in Figure 3, and let $\Sigma = \Sigma_\rightarrow(\mathcal{H}) \cup \{\text{Article} \rightarrow \text{Journal}, \text{Article} \rightarrow \text{Area}, \text{Journal} \rightarrow \text{Area}, \text{Subject}\_\text{Int} \rightarrow \text{Area}, \text{Subject}\_\text{ACM} \rightarrow \text{Area}\}$. Clearly, $\mathcal{D}$ is inconsistent with respect to $\Sigma$. Figure 4 shows several repairs of the dimension $\mathcal{D}$ with respect to $\Sigma$ (we omit in the figure the hierarchy schemas of the repairs since all of them are over the hierarchy schema of $\mathcal{D}$). The repairs are obtained from $\mathcal{D}$ by inserting and/or deleting edges according to the following table:
Figure 4: Repairs of the inconsistent dimension in Figure 3 with respect to $\Sigma$

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Deleted Edges</th>
<th>Inserted Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1$</td>
<td>(DT,MC)</td>
<td>(DT,IS)</td>
</tr>
<tr>
<td>$D_2$</td>
<td>(DKE,H2)</td>
<td>(DKE,G2)</td>
</tr>
<tr>
<td>$D_3$</td>
<td>(DKE,DT)</td>
<td>(DKE,DM)</td>
</tr>
<tr>
<td>$D_4$</td>
<td>(DKE,DT), (H2,IS)</td>
<td>(DKE,DB)</td>
</tr>
<tr>
<td>$D_5$</td>
<td>(DB,IS), (DM,IS), (H2,IS)</td>
<td>(DB,MC), (DM,MC), (H2,MC)</td>
</tr>
<tr>
<td>$D_6$</td>
<td>(DT,MC), (G2,MC), (TODS,DM), (DKE,DT), (A,DKE)</td>
<td>(DT,IS), (G2,IS), (DKE,DB), (A,DKE)</td>
</tr>
</tbody>
</table>

3.1. Notion of Repair

Even though all of the repairs given in Example 8 get rid of the inconsistencies, the repairs with fewer number of changes ($D_1$, $D_2$, $D_3$, $D_4$) are closer to the original dimension than the others. In this section we define minimal repairs, that is repairs that are obtained by applying a minimal number of insertions/deletions. First, we define a notion of distance for dimensions.

**Definition 6.** [Distance between two Dimensions] Given two dimensions $D = (H_D, E_D, Cat_D, <_D)$ and $D' = (H_{D'}, E_{D'}, Cat_{D'}, <_{D'})$, the distance between them, $dist(D, D')$, is defined as $|\langle <_{D'} \setminus <_D \rangle \cup \langle <_D \setminus <_{D'} \rangle|$. 

The distance $dist(D, D')$ is the size of the symmetric difference between the child/parent relations of the two dimensions. Next, we define the notions of repair and minimal repair.
Definition 7. [Repair, Minimal Repair] Let $D = (H_D, E_D, \text{Cat}_D, \prec_D)$ be a dimension and $\Sigma$ be a set of integrity constraints over $H_D$.

(i) A repair of $D$ with respect to $\Sigma$ is a dimension $D' = (H_{D'}, E_{D'}, \text{Cat}_{D'}, \prec_{D'})$ such that $H_D = H_{D'}, E_D = E_{D'}, \text{Cat}_D = \text{Cat}_{D'}$, and $D'$ satisfies $\Sigma$.

(ii) A minimal repair of $D$ with respect to $\Sigma$ is a repair $D'$ such that $\text{dist}(D, D')$ is minimal among all the repairs of $D$ with respect to $\Sigma$. We denote by $\text{MinRepair}(D, \Sigma)$ the set of minimal repairs of $D$ with respect to $\Sigma$.

In the notion of repair we propose, the set of elements in each category of the original dimension is preserved over all the repairs, that is deletions or additions of new elements are not allowed.

Example 9. In order to determine which of the repairs in Figure 4 are minimal, we need to compute their distance to $D$ (Figure 3), which are shown in the following table:

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Symmetric difference with $D$</th>
<th>Distance to $D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1$</td>
<td>$(DT, MC), (DT, IS)$</td>
<td>2</td>
</tr>
<tr>
<td>$D_2$</td>
<td>$(DKE, H_2), (DKE, G_2)$</td>
<td>2</td>
</tr>
<tr>
<td>$D_3$</td>
<td>$(DKE, DT), (DKE, DM)$</td>
<td>2</td>
</tr>
<tr>
<td>$D_4$</td>
<td>$(DKE, DT), (DKE, DB)$</td>
<td>2</td>
</tr>
<tr>
<td>$D_5$</td>
<td>$(DB, IS), (DB, MC), (DM, IS), (DM, MC), (H_2, IS), (H_2, MC)$</td>
<td>6</td>
</tr>
<tr>
<td>$D_6$</td>
<td>$(DT, MC), (DT, IS), (G_2, MC), (G_2, IS), (TODS, DM), (DKE, DT), (DKE, DB), (A_1, TODS), (A, DKE)$</td>
<td>9</td>
</tr>
</tbody>
</table>

Since $D_1, D_2, D_3, D_4$ are closer to $D$, those are the minimal repairs.

Note that we use a cardinality-based repair semantics as opposed to the set-inclusion-based, which is more common in the relational case [6]. This means that we are interested in minimizing the number of changes performed on a dimension, instead of minimizing the set of insertions or deletions of tuples. For example, repairs $D_1$ to $D_5$ in Figure 4 would be minimal repairs under the set inclusion approach, since none of the set differences is a subset of other. Notice that dimension $D_5$ would be a minimal repair even though it is not a good repair. Indeed, it modifies not only the roll-ups of $DKE$ (which is involved in the inconsistencies), but changes the roll-ups of $TODS$ which is not involved in the inconsistencies. Repairs that minimize the number of changes result in more reasonable dimensions in the context of DWs. Cardinality-based relational repairs have been studied in detail in [44, 1].

In the examples shown so far repairs are always obtained by pairs of insertions and deletions. However, this is not always the case, as the following example shows.

Example 10. Consider the hierarchy schema given in Figure 5(a), and the set of constraints $\Sigma = \{A \rightarrow B, A \Rightarrow C\}$. The dimension in Figure 5(b) is inconsistent with respect to $\Sigma$. The inconsistency can be minimally fixed by deleting either $(a, b_1)$ or $(a, b_2)$ and inserting $(a, c)$. These alternatives result in the repairs given in Figures 5(c) and 5(d).

We next prove that there always exists a repair of a dimension that is inconsistent, provided that the dimension has at least one element in each category.

Proposition 1. Let $D = (H_D, E_D, \text{Cat}_D, \prec_D)$, where for each category $c_i \in \text{Cat}_D$, there exists an element $e \in E_D$ such that $c_i = \text{Cat}_D(e)$. Then, there always exists a repair of $D$ with respect to $\Sigma$. 

9
Figure 5: Restoring consistency with respect to the constraints $A \rightarrow B$ and $A \Rightarrow C$

Figure 6: Exponential number of repairs

The next proposition follows directly from the proof of Proposition 1.

**Proposition 2.** Let $D$ be a dimension and $D'$ be a minimal repair of $D$ with respect to a set of constraints $\Sigma$. Then, $\text{dist}(D, D') \leq 2 \times |D|$.

We end this section by noting that the number of minimal repairs could be exponential in the number of elements of the dimension, as the next example shows.

**Example 11.** Consider the dimension given in Figure 6 and the set of constraints $\Sigma = \{A \rightarrow B, A \Rightarrow B\}$. Clearly, the dimension does not satisfy the constraints since each element in category $A$ goes to two different elements in category $B$. A minimal repair is obtained by deleting, for every $a_i$ in $A$, either $a_i < b_1$ or $a_i < b_2$. Thus, there are $2^n$ minimal repairs, where $n$ is the number of elements of category $A$.

3.2. Complexity of Finding Minimal Repairs

Since there can be an exponential number of minimal repairs, it becomes relevant to study the problem of finding one. First, we study the complexity of the decision problem involved.

**Theorem 1.** Let $D$ be a dimension, and $k$ be an integer. The problem of deciding whether there exists a repair $D'$ of $D$ such that $\text{dist}(D, D') \leq k$ is NP-complete.

From Theorem 1, it follows that the problem of computing a minimal repair is NP-hard. Now, we study the complexity of deciding whether a dimension is a minimal repair of another dimension.

**Theorem 2.** Let $D$ be a dimension, and $\Sigma$ be a set of integrity constraints. The problem of deciding whether a dimension $D'$ is a minimal repair of $D$ with respect to $\Sigma$ is co-NP-complete.
The reduction given in the proof of Theorem 1 shows that the problem remains NP-complete even for small schemas. However, the problem can be efficiently solved for dimensions having a single hierarchy path, that is every category has a unique parent category in the hierarchy schema. For example, this happens with the dimension given in Figure 6(a).

**Proposition 3.** Obtaining a minimal repair for a dimension with a single hierarchy path takes polynomial time in the size of the dimension.

### 3.2.1. Repair Distance

As already stated, our main goal is to compute the minimal repairs of an inconsistent dimension with respect to a set of integrity constraints. However, there could be exponentially many repairs, and even if this is not the case, computing the minimal repairs is NP-hard. Despite these negative results, in practical setting, both the number of repairs and the time to compute them are polynomially bounded if we fix the number of insertion/deletions that we need to apply to the original dimension to turn it consistent. Intuitively, this number, we refer to as repair distance, represents the amount of error in the dimension that causes the inconsistency. Formally, the repair distance of the dimension $D$ with respect to $\Sigma$ is the distance between $D$ and a minimal repair of $D$ with respect to $\Sigma$.

**Proposition 4.** Let $D$ be a dimension with $n$ elements, $\Sigma$ be a set of constraints, and $r$ be the repair distance of $D$ with respect to $\Sigma$.

1. The number of minimal repairs of $D$ with respect to $\Sigma$ is in $O(n^2r)$.
2. The minimal repairs $\text{MinRepair}(D, \Sigma)$ can be computed in time $O(n^{2r+3})$.

In practical settings where the errors that cause inconsistencies arise as exceptions in the data, the repair distance of $D$ should be small. As an example, if a dimension has two erroneous edges, which cause all the inconsistencies with respect to a set of constraints, the repair distance is four since just two deletions and two insertions are needed to fix the problem.

### 3.3. Alternative Repair Semantics

When developers or administrators are working on the dimensions trying to solve inconsistencies, they might want to impose more or even less restrictions over the repairs. They might want to make use of their domain knowledge and impose, for example, that certain rollup relations are not modified in the repair process, or that some modifications are preferred over others. On the other hand, if they are not finding a desirable repair among the minimal ones, they might even want to relax the minimality condition.

In what follows, we present several modifications to the repair semantics that could help the user to find the right repairs. Even though the modifications are presented separately, they can be combined if desired.

**Repairing using Ancestor/Descendant Distance.** Even though minimal repairs minimize the changes with respect to the child/parent relation $<$, it is possible that some of them are not minimal in terms of the modifications to the transitive relation of it ($<^*$). In a situation where there are several minimal repairs in terms of $<$, it could be more efficient to explore the ones that also minimize the changes over $<^*$. 

11
For each of the minimal repairs in Example 12, dist minimal among all repairs in MinRepair[Definition 9]. Thus, the repairs that also minimize the ancestor relation of the rollup relations Publication dimension in Figure 1 and suppose that the developer of the DW took the information they might prefer to make no modifications on these relations. As an illustration, consider the Repairing with Preferences. Let D be a dimension, and Σ be a set of constraints Σ, a |-minimal repair D1 of D with respect to Σ is such that D1 ∈ MinRepair(D, Σ), and dist(D, D1) is minimal among all repairs in MinRepair(D, Σ).

Example 12. For each of the minimal repairs in MinRepair(D, Σ) in Figure 4 we have: dist*(D, D1) = [(A2,MC), (A2,IS), (DKE,MC)] = 4.

dist*(D, D2) = [(A2,H2), (A2,G2), (DKE,H2), (DKE,G2), (A2,IS), (A2,MC), (DKE,IS), (DKE,MC)] = 8.

dist*(D, D3) = [(A2,DT), (A2,DM), (DKE,DT), (DKE,DM), (A2,MC), (A2,IS), (DKE,MC), (DKE,IS)] = 8.

Even though all of them are minimal with respect to dist repairs D2, D3 and D4 are at a bigger ancestor/ descendant distance. Therefore, dimension D1 is the only |-minimal repair.

The repair problem is also co-NP-complete in this setting. Indeed membership and hardness follow from the proof in Theorem 2.

Proposition 5. Let D be a dimension, and Σ be a set of integrity constraints. The problem of deciding whether a dimension D is a |-minimal repair of D with respect to Σ is co-NP-complete.

Repairing with Preferences. If users are more confident of some rollup relations than others, they might prefer to make no modifications on these relations. As an illustration, consider the Publication dimension in Figure 1 and suppose that the developer of the DW took the information of the rollup relations R_{df}(Journal, Subject, ACM) and R_{df}(Subject, ACM, Area) from a reliable source, but designed (her)herself relations R_{df}(Journal, Subject, Int) and R_{df}(Subject, Int, Area). In such a case, the developer may prefer to restore consistency by changing the relations (s)he created, but not the ones coming from the more trusted source.

Repairs could be forced to keep some roll-up relations or it could assign costs to to modifications between specific categories and then look for repairs that minimize the total cost of a repair. The complexity of deciding if an instance is a repair in this setting is still co-NP-complete since the semantics introduced in the previous section is a special case.

4. Computing Repairs

The complexity results show that computing repairs of dimensions inconsistent with respect to a set of covering and strictness constraints can be expensive in terms of computational resources. Indeed, unless P equals NP, it is not possible to find tractable implementations for the general case.

However, we can use Datalog programs with negation under stable model semantics with negation and weak constraints [42] to represent and compute the minimal repairs of a dimension. Both the problem of checking if a dimension is a minimal repair and checking if a model is a
stable model of a logic program with negation and weak constraints are co-NP-complete (see Theorem 4.14 in [42]).

Since the repair problem is intractable, any optimal and general solution will be in general inefficient. However, current implementations, such as DLV\(^1\), have extensive optimizations and heuristics to improve efficiency [42]. This logic specification provides a starting point from which other algorithms can be compared in terms of accuracy and efficiency. Further research is needed to address the problem of finding tractable approximate algorithms to find repairs that are not necessarily minimal, and to find specific cases in which the optimal problem is tractable. For instance, in [18] the authors, using the formalizations presented in this work, provide polynomial algorithms for a special type of dimension hierarchy.

In what follows, we will present a Datalog program with negation and weak constraints such that its optimal stable models can be used to obtain the minimal repairs of a dimension with respect to a set \( \Sigma \) of covering and strictness constraints. The repair programs contain rules to keep track of the insertions and deletions of edges needed to restore consistency. Each insertion/deletion operation has a cost associated and the program uses weak constraints to minimize the number of insertions and deletions. The stable models of logic programs with weak constraints are also called optimal stable models since they minimize the number of violations of weak constraints.

In order to simplify the presentation, we present the results in a stepwise manner. We first tackle the case where \( \Sigma = \Sigma_c(\mathcal{H}_D) \cup \Sigma_s(\mathcal{H}_D) \), that is we have to fix a dimension that, according to the constraints, should be strict and covering. Then, we will extend the program to deal with general dimensions where no restrictions are imposed over the constraints.

### 4.1. Repairing to obtain Covering and Strict Dimensions

For a dimension \( D \) inconsistent with respect to \( \Sigma = \Sigma_c(\mathcal{H}_D) \cup \Sigma_s(\mathcal{H}_D) \) we will construct a repair program \( \Pi_{c,s}(D) \) such that there is a one-to-one correspondence between each stable model of it and the minimal repairs of \( D \) with respect to \( \Sigma_c(\mathcal{H}_D) \cup \Sigma_s(\mathcal{H}_D) \). To achieve this, we will use the predicates given in Table 1.

First, let us observe that a covering and strict dimension is such that every element will roll up to exactly one element in each parent category. The repair program relies on this observation and first computes a set of candidate dimensions that contain the same elements as the inconsistent

\[1\]
DLV System: http://www.dbai.tuwien.ac.at/proj/dlv/
dimension an only one rollup between every element and every parent category (note that all this candidate dimensions satisfy the covering constraints). Then, it discards the models that do not satisfy the strictness constraints.

**Definition 10.** [Repair Program for obtaining Covering and Strict Dimensions] The repair program \( \Pi_{\mathcal{C}}(\mathcal{D}) \) for a dimension \( \mathcal{D} = (\mathcal{H}_D, \mathcal{E}_D, \mathcal{C}at_D, <_{D}) \) inconsistent with respect to \( \Sigma_i(\mathcal{H}_D) \cup \Sigma_j(\mathcal{H}_D) \) contains the following rules:

(i) \( C(a) \), for every \( a \in \mathcal{E}_D \) and \( C = \mathcal{C}at(a) \)

(ii) \( R(a, b, c_i, c_j) \), for every \((a, b) \in <_{D} \), with \( c_i = \mathcal{C}at(a) \) and \( c_j = \mathcal{C}at(b) \)

(iii) \( R'(X, Y, c_i, c_j) \leftarrow C(X), C_j(Y), \text{choice}((X, c_i)) \), for every \( c_i \neq c_j \)

(iv) \( R'(X, Y, N_1, N_2) \leftarrow R'(X, Y, N_1, N_2), R'(X, Z, N_1, N_2) \), for every \( c_i \neq c_j \)

(v) \( \leftarrow R'(X, Y, N_1, N_2), R'(X, Z, N_1, N_2), Y \neq Z \).

(vi) \( \text{Ins}(X, Y, N_1, N_2) \leftarrow R'(X, Y, N_1, N_2), \text{not} R(X, Y, N_1, N_2) \).

\( \text{Del}(X, Y, N_1, N_2) \leftarrow R'(X, Y, N_1, N_2), \text{not} R'(X, Y, N_1, N_2) \).

(vii) \( \leftarrow \text{Ins}(X, Y, N_1, N_2)[1 : 1] \).

\( \leftarrow \text{Del}(X, Y, N_1, N_2)[1 : 1] \).

Facts in (i) and (ii) of the repair program correspond to the elements and rollup relations in dimension \( \mathcal{D} \). The rest of the rules in the repair program are used to compute the repairs. In general terms, the program: generates all possible dimensions \( \mathcal{D}' = (\mathcal{H}_{DP}, \mathcal{E}_{DP}, \mathcal{C}at_{DP}, <_{DP}) \) such that \( \mathcal{H}_{DP} = \mathcal{H}_D, \mathcal{E}_{DP} = \mathcal{E}_D, \mathcal{C}at_{DP} = \mathcal{C}at_D \) and such that every element has a unique parent in every parent category (using rules in (iii)), then discards the ones that are not strict (rules in (iv)-(v)) and finally chooses the ones that are at a minimal distance (using rules in (vi)-(vii)).

More specifically, rules in (iii) populate predicate \( R' \) with a new covering child/parent relation for the elements in \( \mathcal{E}_D \). This new relation is generated for every categories \( c_i, c_j \) such that \( c_i \neq c_j \), and every element \( a \) such that \( \mathcal{C}at(a) = c_i \). The rule adds to the repair \( R' \) a child/parent relation of a with a unique element in each parent category \( c_j \). This new element is obtained by using the \text{choice} operator \([28, 29]\) that generates one model for each \( c \) such that \( \mathcal{C}at(c) = c \). In fact, \( \text{choice}((X, c_i), (Y)) \) will assign a unique value to \( Y \) in each stable model for each combination \( (X, c_i) \). The choice operator can be replaced by traditional Datalog rules under the stable model semantics by replacing each rule in (iii) by the following rules:

\( R'(X, Y, c_i, c_j) \leftarrow C_i(X), C_j(Y), \text{chosen}(X, c_j, Y) \).

\( \text{chosen}(X, c_j, Y) \leftarrow C_i(X), C_j(Y), \text{not} \text{diffChoice}(X, c_i, Y) \).

\( \text{diffChoice}(X, c_j, Y) \leftarrow \text{chosen}(X, c_j, Y), C_j(Y), Y \neq Y' \).

After this replacement, in any stable model, the first two attributes of predicate \( \text{chosen}(X, c_j, Y) \) functionally determine the third attribute.

Rules in (iv) compute the transitive closure of the child/parent relation of the repair. Rule (v) is a program constraint (a rule without head) which enforces that it is not possible to have that an element \( x \) rolls-up to an element \( y \), that \( x \) rolls-up to an element \( z \), that \( \mathcal{C}at(y) = \mathcal{C}at(z) \) and that \( y \neq z \). Thus, the rule discards all the models in which the rollup relations, captured in predicate \( RT' \), are not strict.

Rules in (vi) keep track of the insertions and deletions, respectively, that are needed to obtain the repair. The weak constraints \([14]\), denoted by \( \Leftarrow \), in (vii) are used to minimize the number of insertions and deletions needed to restore consistency. In general, a weak constraint is of the form \( \Leftarrow \varphi \ [\text{weight : level}] \), where \( \varphi \) is a conjunction of atoms, and weight and level are integers. If \( \varphi \)
is true in the model the weak constraint is violated and the weight is added up to the cost of the
level. The stable models which minimize the sum of the weights of the violated weak constraints
are called the optimal models of the program. If different levels are used, the optimal models will
minimize first the violation of the constraints of the highest level and then will continue to the
lower levels. In the weak constraints of the repair program both constraint have level = 1 and
weight = 1 since both have to be checked with the same level of priority, and both insertions and
deletions, have the same cost.

Note that rules in (iv) to (vii) are independent of the dimension to repair.

Example 13. Consider the Phone dimension $D_{\text{Phone}}$ in Figure 7. Even though the dimension is
expected to be covering and strict, the dimension is not since the strictness constraint $\text{Number} \rightarrow
\text{Region}$ and the covering constraint $\text{Number} \Rightarrow \text{AreaCode}$ are violated. This is, $D_{\text{Phone}}$ is inco-
herent with respect to the set of constraints $\Sigma_c(H) \cup \Sigma_s(H)$. The repair program $\Pi_{cs}(D)$ contains
the facts in Figure 8, and the following rules:

\[
\begin{align*}
R'(X,Y,\text{number},\text{areaCode}) & \leftarrow \text{Number}(X),\text{AreaCode}(Y),\text{choice}(X,\text{areaCode}(Y)). \\
R'(X,Y,\text{number},\text{city}) & \leftarrow \text{Number}(X),\text{City}(Y),\text{choice}(X,\text{city}(Y)). \\
R'(X,Y,\text{areaCode},\text{region}) & \leftarrow \text{AreaCode}(X),\text{Region}(Y),\text{choice}(X,\text{region}(Y)). \\
R'(X,Y,\text{city},\text{region}) & \leftarrow \text{City}(X),\text{Region}(Y),\text{choice}(X,\text{region}(Y)). \\
R'(X,Y,\text{region},\text{all}) & \leftarrow \text{Region}(X),\text{All}(Y),\text{choice}(X,\text{all}(Y)).
\end{align*}
\]
and Program $\Pi \leftarrow Del \leftarrow Ins \leftarrow RT$

Given a dimension $D$ the sum of weights of each optimal model corresponds to the distance between each of the

**Definition 11.** [Dimension associated to a Model] Given a dimension $D$ and a model $M$ of a repair program for $D$ with respect to $\Sigma$, let the dimension associated to $M$ be $D^M = (H_{D^M}, E_{D^M}, \text{Cat}_{D^M}, c_{D^M})$ with (i) $H_{D^M} = H_D$; (ii) $E_{D^M} = E_D$; (iii) $\text{Cat}_{D^M} = \text{Cat}_D$; and (iv) $c_{D^M} = [(X, Y)] R'(X, Y, c_1, c_j) \in M$.

**Theorem 3.** Given a dimension $D$, every dimension $D^M$ associated to a optimal model $M$ of $\Pi_c(D)$ is a minimal repair of $D$ with respect to $\Sigma_c(H) \cup \Sigma_c(H)$. Furthermore, for every minimal repair $D'$ of $D$ with respect to $\Sigma_c(H) \cup \Sigma_c(H)$ there exists a optimal model $M$ of $\Pi_c(D)$ such that $D^M = D'$.
4.2. Repairing Dimensions: General Case

For a dimension $D$ and a set $\Sigma$ of strictness and covering constraints, we will construct a repair program $\Pi(D, \Sigma)$ such that there is a one-to-one correspondence between each stable model of it and the minimal repairs of $D$ with respect to $\Sigma$. We will use the same predicates as in program $\Pi_{cs}(D)$.

When we were trying to enforce dimensions to be strict and covering, we could take advantage of the fact that every element had only one element in the parent category. In the general case, however, an element might have no parents or as many elements as the parent category. The repair program in this case takes advantage of a different property of minimal repairs.

Example 14. Consider the dimension $D$ in Figure 10 which is inconsistent with respect to $\Sigma = \{A \Rightarrow \text{All}, B \Rightarrow \text{All}, C \Rightarrow \text{All}, D \Rightarrow \text{All}, A \rightarrow C\}$. There are two possible repairs that either delete the child/parent relation $(a_1, b_1)$ or $(a_1, b_2)$ to remove the violation of $A \rightarrow C$ and insert the relation $(a_2, d_1)$ to remove the violation of $A \Rightarrow \text{All}$. The repairs have a distance two and are shown in Figure 10(b)-(c).

For a dimension $D$, an element $a \in E_D$ and a category $c_j \in C_HD$, let $\text{parent}(D', a, c_j) = \{b \mid a <_D b, \text{Cat}(b) = c_j\}$. This is, it contains the set of parents of element $a$ in category $c_j$. The following proposition provides the lower and upper bounds to the number of parents in a category for every element in a minimal repair.

**Proposition 6.** Given a dimension $D$, a set of constraints $\Sigma$ and a minimal repair $D'$ of $D$ with respect to $\Sigma$, for every $a \in E_D$ and $c_j$ such that $\text{Cat}(a) \not\rightarrow_{H_D} c_j$:

$$0 \leq |\text{parent}(D', a, c_j)| \leq \text{Max}\{1, |\text{parent}(D, a, c_j)|\}$$

This relationship between parents in $D$ and every repair $D'$ allows us to put an upper-bound over the size of a repair of $D$. The following corollary follows directly from Proposition 6.

**Corollary 1.** Given a dimension $D$, a set of constraints $\Sigma$ and a minimal repair $D'$ of $D$ with respect to $\Sigma$, the number of child/parent relations in the repair, namely $|<D'|$, is in $O(|<D|)$.

In order to compute the repairs of a dimension $D$ we will construct a new dimension $D_{del}$ from $D$ for which the repairs of $D$ can be computed only through reclassification of child/parent relations of $D_{del}$.
Definition 12. [Del-dimension] Given a dimension \( D \) and a set of constraints \( \Sigma \) let the del-dimension \( D^{del} = (\mathcal{H}_D^{del}, \mathcal{E}_D^{del}, \text{Cat}_D^{del}, <_{\mathcal{D}}^{del}) \) of \( D \) be such that (i) \( \mathcal{H}_D^{del} = \mathcal{H}_D \); (ii) \( \mathcal{E}_D^{del} = \mathcal{E}_D \cup \{ \text{del}_c_i \mid c_i \in \mathcal{C}_{\mathcal{H}_D} \} \); (iii) \( \text{Cat}_D^{del} = \text{Cat}_D \cup \{ \text{del}_c_i \mapsto c_j \mid c_j \in \mathcal{C}_{\mathcal{H}_D} \} \); and (iv) \( <_{\mathcal{D}}^{del} = <_D \cup \{ (a, \text{del}_c_j) \mid a \in \mathcal{E}_D, \text{Cat}_D(a) \uparrow \mathcal{H}_D c_j, \text{parent}(D, a, c_j) = \emptyset \} \).

The del-dimension of a dimension can be used in combination with proposition 6 to give the following result.

Proposition 7. Given a dimension \( D \), a set of constraints \( \Sigma \) and a minimal repair \( D' \) of \( D \) with respect to \( \Sigma \), for every \( a \in \mathcal{E}_D \) and \( c_j \) such that \( \text{Cat}_D(a) \uparrow \mathcal{H}_D c_j \):

\[
0 \leq |\text{parent}(D', a, c_j)| \leq |\text{parent}(D^{del}, a, c_j)|
\]

Starting from \( D^{del} \) we can compute a set of candidate repairs of \( D \) that are obtained by a sequence of reclassifications where each child/parent relation \( (a,b) \in <_{\mathcal{D}}^{del} \) is replaced by a child/parent relation \( (a,c) \) with \( \text{Cat}(c) = \text{Cat}(b) \). Next, the consistency of each candidate repair with respect to the constraints is checked ignoring every child/parent relation with a del-element. Finally, the repairs are obtained from the consistent candidate repairs by removing every del-element.

Example 15. Figure 11(a) shows del-dimension \( D^{del} \) obtained from dimension \( D \) in Figure 10(a) which is inconsistent with respect to \( \Sigma = \{ A \Rightarrow \text{All}, B \Rightarrow \text{All}, C \Rightarrow \text{All}, D \Rightarrow \text{All}, A \Rightarrow C \} \). From it we can compute a set of candidate repairs by reclassifying all the child/parent relations. From these candidate repairs, the only two that satisfy the constraints (ignoring the child/parent relations that involve any del-elements) are shown in Figure 11(b)-(c). If the del-elements are removed, these dimensions correspond to the repairs of \( D \) with respect to \( \Sigma \) shown in Figure 10(b)-(c).

The repair program relies on Proposition 6 and the del-dimension associated to a dimension \( D \). In order to reduce the number of rules and improve efficiency, the repair program uses a single del constant to represent all the del-elements.

Definition 13. [General Repair Program] The repair program for a dimension \( D = (\mathcal{H}_D, \mathcal{E}_D, \text{Cat}_D, <_D) \) with respect to a set \( \Sigma \) of constraints, denoted by \( \Pi(D, \Sigma) \), contains the following rules:

(i) \( C(a) \) for every \( a \in \mathcal{E}_D \) and \( C = \text{Cat}_D(a) \)
The rules in (i)-(iv) construct the dimension $D^{del}$. Indeed, rules in (i) and (ii) represent the elements and rollup relations of dimension $D$, and rules in (iii) and (iv) add the extra $del$-elements and child/parent relations.

Rules in (v) generates the set of candidate repairs by populating predicate $R'$ by reclassifying every child/parent relation in $D^{del}$. Indeed, the rule takes every relation $(X,Y)$ and assigns a new child/parent relation $(X,Z)$ in the repair, where $Z$ is computed with the choose operator that chooses for each pair $(X,Y)$ a different element $Z$ such that $Cat_{D'}(Y) = Cat_D(Z)$ in every repair.

Rules in (vi) are used to compute the descendant/ancestor relationship without considering $del$-elements. This transitive relation is then used to discard the models that violate a strictness or covering constraint. Indeed, for a strictness constraint $c_i \rightarrow c_j$, the rules in (vii) discard the candidate repairs in which an element in $c_i$ has more than one parent in $c_j$. Rules in (viii) are used to discard the models that violate a covering constraint. For example, for a constraint $c_i \Rightarrow c_j$, predicate $aux_{c_j}(X)$ collects elements $X$ for which there exists a child/parent relation to an element in $c_j$ and the models where there is an element in $c_i$ that is not in $aux_{c_j}$ are discarded.

Rules in (ix) keep track of the insertions and deletions, respectively, that are needed to obtain the repair ignoring any modifications that affect $del$-elements. The weak constraints in (x) are used to minimize the number of insertions and deletions needed to restore consistency.

Since this model now has $del$-elements, we need to get rid of them to obtain the repairs. Given a model $M$ of a logic program, let the grounded model of $M$, denoted by $M_{\downarrow}$, be the result of removing from $M$ all the atoms with $del$-elements. Dimension $D^{M_{\downarrow}}$ denotes the dimension of $M_{\downarrow}$.

There are situations in which two different models $M_1$ and $M_2$ might result in the same repair $D'$, this is, $D' = D^{M_{\downarrow}} = D^{M_{\downarrow}}$. Rules in (xii) remove these types of redundancy. Indeed, they restrict the values that the choose operator can assign to discard models that produce duplicate repairs\(^2\). The first rule discards models in which child/parent relations are switched resulting in

\(^2\)Note that the predicate $chosen(x,y,z,w)$ is used in the Datalog rules that replace the choice operator. It represents the reclassification of element $x$ from $y$ to $w$ in category $z$. 

19
the same child/parent relations. For example, if an element a rolls-up to both b and c, a model which reclassifies (a,b) to (a,c) and (a,c) to (a,b) results in no modification to the associated dimension and therefore should be discarded. The second rule discards the models in which two child/parent relations (a,b) and (a,c) are reclassified to the same element d. This reclassification produces the same result as the model in which (a,b) (or (a,c)) is reclassified to d and (a,c) (or (a,b)) is reclassified to del. Finally, the third rule discards models in which a child/parent relation (a,c) is deleted by reclassifying it to del, but reclassifies (a,b) back to c. This model is equivalent to one where (a,c) is not modified and (a,b) is deleted by reclassifying to del. These rules are not needed to ensure correctness and completeness of program $\Pi(\mathcal{D}, \Sigma)$ but improve efficiency by reducing the computation of duplicate models.

**Example 16.** For the dimension $\mathcal{D}$ in Figure 10 which is inconsistent with respect to $\Sigma = \{A \rightarrow \text{All}, B \rightarrow \text{All}, \text{C} \rightarrow \text{All}, \text{D} \rightarrow \text{All}, \text{A} \rightarrow \text{C}\}$, the repair program $\Pi(\mathcal{D}, \Sigma)$ contains the facts in Figure 12, and the following rules:

\begin{align*}
A(\text{del}). & \Rightarrow \text{All(\text{del}).} \\
B(\text{del}). & \Rightarrow \text{R(a₁, \text{del}, A, D).} \\
C(\text{del}). & \Rightarrow \text{R(a₂, \text{del}, A, D).} \\
D(\text{del}). & \Rightarrow \text{R(\text{del}).}
\end{align*}

\begin{align*}
R'(X, Y', Z, B) & \leftarrow R(X, Y, Z, B), B(Y'). \\
R'(X, Y', Z, C) & \leftarrow R(X, Y, Z, C), C(Y'). \\
R'(X, Y', Z, D) & \leftarrow R(X, Y, Z, D), D(Y'). \\
R'(X, Y', Z, \text{All}) & \leftarrow R(X, Y, Z, \text{All}), \text{All}(Y').
\end{align*}

\begin{align*}
& \Rightarrow R'(X, Y, A, C), R'(X, Z, A, C), Y ≠ Z. \\
& \leftarrow A(X), \text{not aux}_{\text{del}}(X), X ≠ \text{del}. \\
& \leftarrow B(X), \text{not aux}_{\text{del}}(X), X ≠ \text{del}. \\
& \leftarrow C(X), \text{not aux}_{\text{del}}(X), X ≠ \text{del}. \\
& \leftarrow D(X), \text{not aux}_{\text{del}}(X), X ≠ \text{del}. \\
& \text{aux}_{\text{del}}(X) \leftarrow R'(X, Y, C, 1), \text{All}.
\end{align*}

\begin{align*}
\text{Ins}(X, Y, N₁, N₂) & \leftarrow R'(X, Y, N₁, N₂), \text{not } R(X, Y, N₁, N₂), Y ≠ \text{del}. \\
\text{Del}(X, Y, N₁, N₂) & \leftarrow R'(X, Y, N₁, N₂), \text{not } R(X, Y, N₁, N₂), Y ≠ \text{del}. \\
& \equiv \text{Ins}(X, Y, N₁, N₂)[1 : 1] \\
& \equiv \text{Del}(X, Y, N₁, N₂)[1 : 1]. \\
& \leftarrow \text{chosen}(X, Y', N, Y'), \text{chosen}(X, Y', N, Y'), Y ≠ Y'. \\
& \leftarrow \text{chosen}(X, Z₁, N, Y), \text{chosen}(X, Z₂, N, Y), Z₁ ≠ Z₂, Y ≠ \text{del}. \\
& \leftarrow \text{chosen}(X, Y, N, \text{del}), \text{chosen}(X, Y', N, Y), Y ≠ Y'.
\end{align*}

Program $\Pi(\mathcal{D}, \Sigma)$ has two best models $M₁ = \{\text{Ins}(a₂, d₁, A, D), \text{Del}(a₁, b₂, A, B)\}$ and $M₂ = \{\text{Ins}(a₂, d₁, A, D), \text{Del}(a₁, b₁, A, B)\}$, both at distance 2. The dimensions associated to these models are the repairs of $\mathcal{D}$ as shown in Figure 10(b)-(c).

**Theorem 4.** Given a dimension $\mathcal{D}$ and a set of constraints $\Sigma$, for every best model $M$ of $\Pi(\mathcal{D}, \Sigma)$ the dimension $\mathcal{D}^{M₁}$ is a minimal repair of $\mathcal{D}$ with respect to $\Sigma$. Furthermore, for every minimal repair $\mathcal{D}'$ of $\mathcal{D}$ with respect to $\Sigma$ there exists a best model $M$ of $\Pi(\mathcal{D}, \Sigma)$ such that $\mathcal{D}^{M} = \mathcal{D}'$. □

The efficiency of the implementation can be improved by using different types of repair rules for every pair $cᵢ ↗ cⱼ$ depending on the constraints that apply between them. For example, if the
relation between \( c_i \) and \( c_j \) is expected to be both strict and covering, we can use the rules defined in Section 4.1 which are more efficient than the ones in the general case. Another special case in which the program can be simplified is when a covering dimension is inconsistent with respect to \( \Sigma \) with \( \Sigma \subseteq \Sigma_r(\mathcal{H}) \). In this case \( 1 \leq |parent(\mathcal{D}', a, c_j)| \leq |parent(\mathcal{D}, a, c_j)| \), and therefore the repairs can be obtained with reclassifications but without the del-elements. This is the case with the publication dimension.

### 4.3. Alternative Semantics

The logic programs can be easily modified to consider the alternative repair semantics introduced in Section 3.3.

**Repairing using Ancestor/Descendant Distance.** Program \( \Pi(\mathcal{D}, \Sigma) \) in Definition 13 can be modified to compute \(<^*\)-minimal repairs by adding the following rules:

- (xii) \( RT(x, y, n_1, n_2) \leftarrow R(x, y, n_1, n_2), X \neq \text{del}, Y \neq \text{del} \).
- (xiii) \( RT(x, z, n_1, n_3) \leftarrow R(x, y, n_1, n_2), RT(y, z, n_2, n_3), X \neq \text{del}, Y \neq \text{del}, Z \neq \text{del} \).
- (xiv) \( \text{Ins}(x, y, n_1, n_2) \leftarrow RT'(x, y, n_1, n_2), \neg RT(x, y, n_1, n_2) \).
- (xv) \( \text{Del}(x, y, n_1, n_2) \leftarrow RT(x, y, n_1, n_2), \neg RT'(x, y, n_1, n_2) \).
- (xvi) \( \text{Ins}(x, y, n_1, n_2) \). [1 : 2]
- (xvii) \( \text{Del}(x, y, n_1, n_2) \). [1 : 2]

Rules in (xii) and (xiii) populate \( RT \) with the transitive closure of \( R \). Rules in (xiv) and (xv) compute the edges that have been inserted and deleted from \( RT \) to generate \( RT' \). The weak constraints control the number of changes made to the transitive closure of the rollup relation.

Here, the level of these weak constraints is two, since we first want to minimize changes made to relation \( R \) and after that to minimize the changes made to \( RT \).

**Repairing with Preferences.** Preferences can be expressed in the repair programs by using weak constraints with different weights. For example, we can assign a cost of three (3) to insertions and deletions between categories \( n_1 \) and \( n_2 \) and of four (4) between \( n_3 \) and \( n_4 \).

- \( \text{Ins}(x, y, n_1, n_2) \). [3 : 1]  
- \( \text{Del}(x, y, n_1, n_2) \). [3 : 1]  
- \( \text{Ins}(x, y, n_1, n_2) \). [4 : 1]  
- \( \text{Del}(x, y, n_1, n_2) \). [4 : 1]

Thus, it is possible to assign different costs to modifications between specific categories or specific elements. It is also possible to use the different priority levels of weak constraints and therefore, allowing to further refine the preferences.

Furthermore, using traditional constraints and facts the developer could force the repair program to keep some edges in the repairs. For instance, a user may prefer to forbid modifications...
of edges between two specific categories \( c_1 \) and \( c_2 \). This restriction can be specified in the repair program with the following constraints:

\[
\textsf{Ins}(X, Y, c_1, c_2),
\]

\[
\textsf{Del}(X, Y, c_1, c_2).
\]

Moreover, we can force some edges to be kept in the repairs. This can be achieved by inserting facts to predicate \( R' \) of the repair program. For instance, if we insert the fact \( R'(a,b,c_1,c_2) \) into a repair program, then the edge \((a,b)\) will be in every repair generated by it.

Finally, if a user prefers to relax the minimality condition, (s)he can request the models with a cost smaller than \( N \) simply by running DLV with the option \(-\text{costbound}=N\). For instance, if we ask for the models with cost less than 7 we will obtain repairs \( D_1, \ldots, D_5 \) in Figure 4.

5. Related Work

**Inconsistencies in DWs.** Since DWs are conceived as collections of materialized views of data extracted from operational databases, much effort has been centered on the problem of resolving inconsistencies between operational databases and DWs [26, 32, 58, 59, 61, 39]. Other works have studied imprecise or missing data [54, 15] in dimensions. In [15] the multidimensional data model is extended to support data ambiguity, imprecision, and uncertainty. In [47] a conceptual multidimensional model is extended with temporal features.

Few works have tackled the problem of resolving the inconsistencies that arise in DW dimensions. In early research on DWs, dimensions were consider the static part of DWs. Further work [37, 36] has shown that dimensions need to be adapted due to changes in data sources or the evolution of business rules. When DWs are updated, dimensions may become inconsistent with respect to their constraints. In addition, inconsistency may be caused by disparity across the different data sources that feed a DW, or by erroneous or noisy data.

Letz et al. [43] motivate the importance of enforcing strictness constraints in dimensions using constraints represented in dimensions hierarchies. In particular, the constraints are used to keep dimensions strict upon successive update operations. Pedersen et al. [53] present a method to transform non-strict dimensions into strict dimensions. In this method, strictness is restored by inserting new artificial elements into some categories. As an example, if an element \( a \) rolls-up to both \( b \) and \( c \) in the same category, a new element \((b,c)\) is created and \( a \) is associated with this new element. The method proposed by Pedersen et al. [53] is useful when the data is correct but do not conform to the strictness restriction. As an example, a river which is in the border of two countries may rollup to two elements in the same country category. In this case, the solution is not to repair the dimension (in our sense), but to transform it to conform the restrictions of the multidimensional model. In contrast, our method is aimed at supporting the cleaning of errors that turn a dimension inconsistent with respect to a set of constraints.

**Inconsistent Relational Databases.** The problem of repairing relational databases with respect to a set of integrity constraints (e.g. functional dependencies and inclusion dependencies) has been extensively studied (see [6] for a survey).

Even though there are several ways to represent a data warehouse using relational models (e.g. star schema or snowflake schema [19, 21, 40]) it is not possible to use the relational repair techniques to compute DW’s repairs. This is because, the minimal relational repairs obtained by tuple deletions, insertions or updates do not coincide with the minimal repairs of DWs. Indeed, they might result in the removal of more roll-up relations than needed or in discarding repairs.
that are minimal in the DWs sense, but not in the relational case. As an illustration, consider the Phone dimension $D_{\text{Phone}}$ in Figure 13(a) and the set of constraints $\Sigma = \{\text{Number} \rightarrow \text{Region}, \text{AreaCode} \Rightarrow \text{Region}, \text{City} \Rightarrow \text{Region}, \text{Number} \Rightarrow \text{AreaCode}\}$. The dimension is inconsistent with respect to $\Sigma$ since number $N_1$ reaches two different regions. There is a unique repair at distance 1 that is obtained by deleting edge $(N_1, \text{TCH})$ (this is possible since the rollup relation between categories Number and City is not required to be covering).

Figure 13(b) is the relational representation of this dimension following the star schema. The functional dependency $\text{Number} \rightarrow \text{Region}$ over table $\text{Phone}$ enforces the strictness constraint $\text{Number} \rightarrow \text{Region}$. According to the relational approach, where functional dependencies are repaired by tuple deletions, the repairs for this dimension are obtained by deleting tuple $(N_1, 45, \text{TCH}, \text{VIII})$ or tuple $(N_1, 45, \text{TCH}, \text{IX})$. None of the relational repairs correspond to the minimal repair in the DW sense. The situation does not change if we use a snowflake schema as we illustrate below.

Let us assume that the Phone dimension is covering. The following tables are the snowflake representation of the Phone dimension in Figure 13(a):

<table>
<thead>
<tr>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>AreaCode</td>
<td>Number</td>
<td>City</td>
</tr>
<tr>
<td>$N_1$</td>
<td>45</td>
<td>$N_1$</td>
<td>TCH</td>
</tr>
<tr>
<td>$N_2$</td>
<td>45</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The strictness constraint $\text{Number} \rightarrow \text{Region}$ can be represented with the constraint:

$$\forall xyz_1z_2w (T_1(x,y), T_2(x,w), T_3(y,z_1), T_4(w,z_2) \rightarrow z_1 = z_2)$$

The following constraints enforce the covering condition: $\forall x (\text{Number}(x) \rightarrow \exists w T_1(x,w)), \forall x (\text{Number}(x) \rightarrow \exists y T_2(x,y), \forall x (\text{AreaCode}(x) \rightarrow \exists y T_3(x,y))$ and $\forall x (\text{City}(x) \rightarrow \exists z T_4(x,z))$. The dimension is inconsistent since element $N_1$ reaches regions IX and VIII. There are two minimal repairs (in our sense). The first one is obtained by deleting edge (45,IX) and inserting (45,VIII). The second one is obtained by deleting edge (TCH,IX) and inserting (TCH,VIII). In the relational representation consistency can be restored by deleting tuples in any of the relations representing the dimension’s hierarchy. In particular, if we delete tuple $T_2(N_1,\text{TCH})$ we need to introduce tuple $T_2(N_1,\text{NULL})$ in order to keep consistency with respect to $\text{Number}(x) \rightarrow \exists v T_2(x,v)$ (see [11] for restoring consistency of foreign key in relational databases). Note that, this change does not generate a repair in our sense.

23
Another important difference with the classical relational setting is that, in the case of DWs we use a cardinality-based repair semantics [44, 1] as opposed to the set-inclusion-based, which is more common in the relational case [6]. This means that we are interested in minimizing the number of changes performed on a dimension, instead of minimizing the set of insertions or deletions of tuples.

**Data cleaning.** There is a lot of research in the area of data cleaning (see [5] for a survey) but the one closest to our problem corresponds to data cleaning based on integrity constraints (see [24]). In [20] a framework to improve data quality with respect to conditional functional dependencies is presented. There, database repairs are obtained by attribute modifications. To improve the quality of repairs, authors implement an statistical method that modifies attribute values and ensures that the repairs are accurate, this is, closer to the “correct” data. Their techniques are not directly applicable to repair inconsistent dimensions in DWs, but they are an interesting line for future research. In [23] authors address the issue of repairing databases that are inconsistent with respect to a set of conditional dependencies, an extension of functional and inclusion dependencies. The repairs are also obtained by performing attribute value modifications.

**Repair Logic Programs.** Logic programs work as a compact representation of database repairs. They have been used to specify the repairs of relational databases with respect to relational integrity constraints (e.g. functional dependencies, foreign key constraints). In particular, in [3, 4, 11], database repairs are specified as the stable models of disjunctive logic programs with stable model semantics [27] (also called *answer set programs*). Logic-based approaches for repairing data integration systems are presented in [31, 12, 22], and for repairing and querying inconsistent peer-to-peer systems in [7, 17].

### 6. Conclusion

Since an enterprise data warehouse can contain terabytes of data [19, 60] ensuring that the dimensions satisfy the required integrity constraints can be vital for an efficient computation of reports and summaries. Therefore, it is important to develop tools to allow the designer of a DW to check integrity constraints and to help in the process of repairing inconsistencies with respect to them. We believe that such tools would be particularly useful in the creation and cleaning stage of DWs and when the dimensions are updated and adapted due to changes in data sources or modifications to the business rules [37, 36, 49, 10].

In this paper, we propose a method to support the restoring of consistency of DW dimensions with respect to a set of constraints. In particular, we provide formalization of the dimension (minimal) repairs with respect to covering and strictness constraints. We present a complexity analysis of the problem of computing a minimal repair, and show that in general the problem is NP-hard. However, we show an specific case in which it can be done in polynomial time. We also prove that repairs always exist. We have explored a novel line of research which is the use of Datalog programs with stable model semantics with weak constraints [42] to represent database repairs and extensions to the repair semantics. These logic programs solve NP-complete problems, and therefore they allow to compute repairs of general dimensions, but their evaluation over large sets of data will be in general inefficient. However, they provide a starting point from which other, probably approximate algorithms, can be compared for quality and efficiency. Thus, it is relevant to investigate other kind of algorithms to compute repairs as in [18], where polinomial
time algorithms are presented to compute dimension repairs (in our sense) for a special class of data warehouse’s dimensions.

Repairs of dimensions are obtained by performing insertions and deletions of edges between dimension elements. We do not consider the insertion or elimination of dimension elements, options that might be helpful to restore consistency of dimensions. We leave this analysis for future work. We suggest alternative repair semantics that take into consideration user preferences such as avoiding the modification of some roll-up relations, assigning different weights to changes in different relations, etc. We also leave for future work the definition of repairs that satisfy some accuracy measure as in [20].

Another interesting idea for future research is to compute consistent answers to queries over DWs, following the framework proposed in [2, 9, 6], where repairs are used as auxiliary concepts to characterize consistent answers to queries. A preliminary analysis in this direction is presented in [8].

Acknowledgements: Mónica Caniupán was funded by FONDECYT grant number 11070186, and DIUBB 110115 2/R. Loreto Bravo was funded by CONICYT grant PSD-57. Carlos Hurtado was partially funded by FONDECYT grant number 1080672.

References


Proof of Proposition 1: It is enough to construct a repair $D' = (\mathcal{H}_D, \mathcal{E}_D, \text{Cat}_D, \prec_D)$ of $D$ with respect to $\Sigma$ ($D'$ is not necessarily a minimal repair). First, for each category $c \in C_D$ we choose a specific element $e^c \in E_D$ such that $c_i = \text{Cat}_D(e^c)$ (it does not matter which element) and let $\prec_D = \emptyset$. Then, for all categories $c_i$ and $c_j$ such that $c_i \rightarrow c_j \in \Sigma$ and for every $a \in e_i$ we add $(a, e^c)$ to $\prec_D$. The dimension obtained is a repair since, by construction, it satisfies the covering constraints, and it is strict since in each category there is a unique element to which all the elements in lower categories roll up. For example, repair $D_b$ given in Figure 4 is obtained in this way. \hfill $\Box$

Proof of Theorem 1: Membership in NP is direct. Nondeterministically choose a dimension $D'$ and test in polytime whether $\text{dist}(D, D') \leq k$.

In order to prove NP-hardness, we present a reduction from the set covering problem which is known to be NP-complete [51]. The input of the set covering problem is a set $\mathcal{U} = \{u_1, \ldots, u_n\}$, a set $\mathcal{S} = \{s_1, \ldots, s_m\}$ where each $s_i \subseteq \mathcal{U}$, and an integer $k$. The question is whether there is a set covering for $\mathcal{U}$, $S$ of size $k$ or less (a covering for $\mathcal{U}$, $S$ is a set $C \subseteq \mathcal{S}$ such that $\mathcal{U} = \bigcup_{s \in C} s_i$).

For a set covering problem with $\mathcal{U} = \{u_1, \ldots, u_n\}$, $\mathcal{S} = \{s_1, \ldots, s_m\}$ and $k$, we construct a dimension $D = (\mathcal{H}_D, \mathcal{E}_D, \text{Cat}_D, \prec_D)$ (see Figure 14) and a set of constraints $\Sigma$. The hierarchy schema $\mathcal{H}_D = (\mathcal{C}_D, \succ_D)$ is as follows:

- $C_{\mathcal{H}_D} = \{U, T, S, R, \text{All}\}$; $\forall \mathcal{C}_D \text{ All}$; and
- $\succ_{\mathcal{H}_D} = \{(U, T), (U, S), (T, R), (S, R), (R, \text{All})\}$.

The rest of $\mathcal{D}$ is as follows:

- $\mathcal{E}_D = \{u_1, \ldots, u_n, l, s_1, \ldots, s_m, r, r_1, \ldots, r_m, \text{ all}\}; \text{ all}_D = \text{ all}$;
- $\text{Cat}_D = \{u_1 \mapsto U, \ldots, u_n \mapsto U, t \mapsto T, \mathcal{S}_i \mapsto S, \ldots, s_m \mapsto S, r \mapsto R, r_1 \mapsto R, \ldots, r_m \mapsto R, \text{ all} \mapsto \text{ All}\}$; and
- $\prec_D = \{(u_1, l), \ldots, (u_n, l), (l, t), (s_1, r_1), \ldots, (s_m, r_m), (r, \text{ all}), (r_1, \text{ all}), \ldots, (r_m, \text{ all})\} \cup \{u, s \} | u \in \mathcal{U}, s \in \mathcal{S}\}$, and $u_i \in s_i$.

Figure 14: Dimension $D$ for the reduction

The set of constraints $\Sigma$ contains $\Sigma(\mathcal{H}_D) \cup \Sigma(\mathcal{H}_D)$, that is, for each edge from $c_i$ to $c_j$ in $\mathcal{H}_D$, $\Sigma$ has a covering constraint $c_i \rightarrow c_j$ and a strictness constraint $c_i \Rightarrow c_j$. 28
Notice that $D$ may be inconsistent with respect to $\Sigma$.

Next, we prove that (A) there is a set covering $C$ for $U, S$ of size $k$ or less if and only if (B) there is a repair $D'$ of $D$ with respect to $\Sigma$ at distance $2k + q$ or less, where $q = (\sum_{s_i \in S} |s_i|) - |U|$ (without loss of generality we may assume that $q \geq 0$, otherwise the set covering problem can be straightforwardly solved in polytime).

The only if part is direct. Assuming (A), let $C$ be the set covering such that $|C| \leq k$, we obtain a repair $D'$ from $D$ as follows. Initially let $<_{D'} = <_{D}$. Now, we (i) delete from $<_{D'}$ all pairs $(s_i, t_i)$ in $<_{D}$ such that $s_i \in C$. Then, we (ii) add to $<_{D'}$ all pairs $(s_i, t_i)$ such that $s_i \in C$. Then, we (iii) delete from $<_{D'}$ all the pairs $(u_i, s_j)$ such that $s_j \notin C$. Finally, (iv) for every element $u_i$ we delete from $<_{D'}$ all the pairs $(u_i, s_j)$ but one. Notice that steps (i) and (ii) imply $|C|$ deletions and $|C|$ additions. Steps (iii) and (iv) comprise $q$ deletions. Also, notice that $D'$ satisfies the constraints $\Sigma$. Therefore, $D'$ is repair of $D$ with respect to $\Sigma$ at distance $2|C| + q \leq p$.

For the if part, assuming (B), let $D'$ be the repair of $D$ at distance $d' \leq 2k + q$.

Now, assume that $(t, r) \notin <_{D'}$. Let $t_i$ be the element in $R$ such that $(t, t_i) \in <_{D'}$. We apply the following updates to $D'$: (i) delete $(t, t_i)$ and add $(t, r)$; and (ii) for all pairs $(s_j, t_i) \in <_{D'}$, delete $(s_j, t_i)$ and add $(s_j, r)$. It can be easily verified that after applying the aforementioned updates $D'$ is consistent with respect to $\Sigma$ and $\text{dist}(D, D') = d'$. Therefore, without loss of generality we can assume that $(t, r) \in <_{D'}$.

Next, we prove that, assuming (B), there is always a repair $D''$ of $D$ at distance $d'' \leq 2k + q$ such that $\mathcal{R}_{D''}(U, S) \subseteq \mathcal{R}_{D'}(U, S)$. We construct $D''$ from $D'$ as follows: initially we let $<_{D''} = <_{D'}$.

Then, for each pair $(u_i, s_j) \in <_{D'}$ such that $(u_i, s_j) \notin <_{D'}$ we perform the following updates: (i) choose an edge $(u_i, s_j) \in <_{D'}$, then delete $(u_i, s_j)$ from $<_{D''}$ and add $(u_i, s_j)$ to $<_{D''}$. (ii) if it is not the case that $(s_j, t_i) \in <_{D''}$, that is, there exists an element $t_p$ such that $(s_j, t_p) \in <_{D''}$ and $t_p \neq t$, then delete the edge $(s_j, t_p)$ from $<_{D''}$ and add the edge $(s_j, r)$ to $<_{D''}$.

In the updates done in step (i) $\text{dist}(D'', D')$ is decreased by 2, and in the updates done in step (ii) $\text{dist}(D'', D)$ is increased at most by 2. Therefore, $d'' \leq d' \leq 2k + q$. Notice that $D''$ is consistent with respect to $\Sigma$.

Let $C$ be the set containing every element $s_j$ in category $S$ such that there exists an element $u_i$ in category $U$ and $u_i <_{D''} s_j$. Observe that $d'' \geq 2|C| + q$, because the edges $(s_j, t)$ do not belong to $<_{D''}$, and there are $|C|$ of such edges, and $|\mathcal{R}_{D''}(U, S) \setminus \mathcal{R}_{D'}(U, S)| = q$. Also, it can be verified that $C$ is a cover for $\mathcal{U}, S$. Now, assume that $|C| > k$. Then $d'' < 2|C| + q$, a contradiction.

As an illustration of the reduction given in this proof, suppose we have a set $\mathcal{S}$ of sets $s_1, s_2, s_3$ with domain $\mathcal{U} = \{a, b, c\}$, such that $S_1 = \{a, b\}$, $S_2 = \{c\}$, and $S_3 = \{a, c\}$. The minimal covers are $\{s_1, s_2\}$ and $\{s_1, s_3\}$. We generate the dimension $D$ shown in Figure 14(c) for $n = 3$ and $m = 3$.

Proof of Theorem 2: Consider the complement problem of deciding whether $D'$ is not a minimal repair of $D$ with respect to $\Sigma$. We have that $D'$ is not a minimal repair of $D$ with respect to $\Sigma$ if and only if (*) there is a repair $D''$ of $D$ with respect to $\Sigma$ at distance $\text{dist}(D, D'') \leq k = \text{dist}(D, D')$. And from Theorem 1, it follows that (*) is NP-complete.

Proof of Proposition 3: Considering an element $o$ in some category. If the element cause the violation of a strictness constraint, we just delete from the child/parent relation all the edges but one, that start from the element. If $o$ causes the violation of a covering constraint, we just add one edge that start from $o$ to the child/parent relation. This procedure takes $O(n^2)$ steps, where $n$ is the number of elements in the original dimension.

29
Proof of Proposition 4:
1. The number of edges of a dimension is in $O(n^2)$. A repair at distance $k$ is obtained by applying $k$ insertions/deletions. Each insertion/deletion can be applied over $O(n^2)$ possible edges.

   Therefore in order to select $k$ insertions/deletions we have to make $O(n^{2k})$ choices. Now, for $k = r$ we obtain the result.

2. From (1) it follows that we have to check the consistency of $O(n^{2r})$ steps. $\square$

Proof of Theorem 3: Correctness. Rule $R'(X, Y, c_i, c_j) \leftarrow C_i(X), C_j(Y)$, choice((X, c_i)(Y)) generates all the possible combinations of pairs $(X, c_i)$ and $(Y, c_j)$, this is, it will generate all the possible stable models in which an element $X$ has a unique parent $Y$ in category $c_j$. This is direct consequence of the use of the choice operator and its properties [28].

Let $\text{CandRepair} = \{D|\text{M} \text{ is a stable model of rules (i)-(iii)}\}$. Every repair $D'$ of dimension $D$ with respect to $\Sigma,(H) \cup \Sigma,(H)$ is such that $D' \in \text{CandRepair}$.

By construction every $D' \in \text{CandRepair}$ is covering. Thus, we need to discard the models that do not satisfy the strictness constraints. This is achieved with rules in (iv) and (v). Let us assume by contradiction that a dimension $D^M$ associated to a stable model of $\Pi_{\Sigma}(D)$ is not strict. This implies that there exists $c_i, c_j \in C_{H_0}$ and elements $a, b_1, b_2 \in E_D$ such that (i) $c_i \neq H_0 c_j$, (ii) $\text{Cat}_{\Sigma}(a) = c_i, \text{Cat}_{\Sigma}(b_1) = c_i$, and $\text{Cat}_{\Sigma}(b_2) = c_j$; and (iii) $a <_{\Sigma} b_1$ and $a <_{\Sigma} b_2$. By construction of $D^M$ and transitivity properties that $\text{M}$ contains $R'(a, b_1, c_i, c_j)$ and $R'(a, b_2, c_i, c_j)$. These atoms would violate rules in (v). Thus, this is not possible and rules in (iv) and (v) discard all the candidate repairs in $\text{CandRepair}$ that are not strict.

So far we have proven that the dimensions associated to the stable models of a program containing rules (i) to (v) are both covering and strict. In other words, they are all repairs of $D$ with respect to $\Sigma,(H) \cup \Sigma,(H)$. What we are missing now is to discard the models for which the associated dimensions are repairs which are not minimal. This is achieved through the weak constraints that were proven [42] to minimize the number of times atoms are made true. In this case, they will minimize the number of Ins and Del atoms in the optimal stable models. This is, they will keep the optimal models among the stable models of the program.

Completeness. For a repair $D' = (H_D, E_D, \text{Cat}_{\Sigma}, <_{\Sigma})$ of $D = (H_D, E_D, \text{Cat}_{\Sigma}, <_{\Sigma})$ with respect to $\Sigma,(H) \cup \Sigma,(H)$ we will construct a stable model $M'$ such that $D^M = D'$ that is optimal. Let $M'$ be the union of the following sets:

- $\{C(a)|a \in E_D \text{ and } C = \text{Cat}(a)\}$
- $\{R(a, b, c_i, c_j)|(a, b) \in <_{\Sigma} c_i = \text{Cat}(a) \text{ and } c_j = \text{Cat}(b)\}$
- $\{R'(a, b, c_i, c_j)|(a, b) \in <_{\Sigma} c_i = \text{Cat}(a) \text{ and } c_j = \text{Cat}(b)\}$
- $\{RT'(a, b, c_i, c_j)|(a, b) \in <_{\Sigma} c_i = \text{Cat}(a) \text{ and } c_j = \text{Cat}(b)\}$
- $\{\text{Ins}(a, b)|(a, b) \in <_{\Sigma} \text{ and } (a, b) \notin <_{\Sigma}\}$
- $\{\text{Del}(a, b)|(a, b) \in <_{\Sigma} \text{ and } (a, b) \notin <_{\Sigma}\}$

The dimension associated to $M'$ is precisely $D'$. Also, $M'$ is a stable model of $\Pi_{\Sigma}(D)$. Finally, since $D'$ is a minimal repair, it is easy to check that $M'$ is also a optimal model. $\square$

Proof of Proposition 6: Sketch. Given a dimension $D$, a set of constraints $\Sigma$ and a minimal repair $D'$ of $D$ with respect to $\Sigma$, for every $c_i, c_j$ with $c_i \neq H_0 c_j$ and every $a \in E_D$ with $\text{Cat}_{\Sigma}(a) = c_i$:

1. If $\{c_i \rightarrow c_j, c_j \Rightarrow c_i\} \subseteq \Sigma$, then $|\text{parent}(D', a, c_j)| = 1$.
2. If $\{c_i \rightarrow c_j\} \subseteq \Sigma$, then $0 \leq |\text{parent}(D', a, c_j)| \leq 1$.
3. If $\{c_i \Rightarrow c_j\} \subseteq \Sigma$ and $|\text{parent}(D, a, c_j)| = 0$, then $|\text{parent}(D', a, c_j)| = 1$. 

30
4. If \( \{c_i \Rightarrow c_j\} \in \Sigma \) and \( |\text{parent}(D, a, c_j)| \geq 1 \), then \( 1 \leq |\text{parent}(D', a, c_j)| \leq |\text{parent}(D, a, c_j)| \).
5. If there are no constraints for \( c_i \) and \( c_j \), and \( |\text{parent}(D, a, c_j)| = 0 \), then \( 0 \leq |\text{parent}(D', a, c_j)| \leq 1 \).
6. If there are no constraints for \( c_i \) and \( c_j \), and \( |\text{parent}(D, a, c_j)| \geq 1 \), then \( 0 \leq |\text{parent}(D', a, c_j)| \leq |\text{parent}(D, a, c_j)| \).

**Proof of Theorem 4:** Correctness. Rules (i)-(iv) construct the logical representation of the elements and child/parent relations of the \( \delta \)-dimension \( D^{\delta M} \) of \( D \).

Rule \( R'(X, Y', Z, c_j) \leftarrow R(X, Y, Z, c_j), C_j(Y') \), choice \((X, Y, c_j)(Y')\) generates all the possible combinations of pairs \((X, Y, c_j)\) and \((Y')\), this is, it will generate all the possible stable models in which a child/parent relation \( X \prec_\delta Y \) is reclassified to \( Y' \) in category \( c_j \). This is a direct consequence of the use of the choice operator and its properties [28].

Let the set of candidate repairs \( \text{CandRepair} = \{ D^{\delta M} | M \} \) is a stable model of rules (i)-(iii)]. By the properties of the choice operator and the removal of atoms with \( \delta \)-elements, for every \( D^{\text{and}} \in \text{CandRepair}, a \in E_D \) and \( c_j \) such that \( \text{Cat}_D(a) \not\sim_{H_0} c_j \): \( 0 \leq |\text{parent}(D^{\text{and}}, a, c_j)| \leq \text{Max}[1,|\text{parent}(D, a, c_j)|] \). Furthermore, every repair \( D' \) of dimension \( D \) with respect to \( \Sigma \) is such that \( D' \in \text{CandRepair} \).

Rules (vi)-(viii) discard the models from \( \text{CandRepair} \) that violate a covering or strictness constraint. In a stable model of the program rules in (vi) compute the transitive closure of \( R' \) ignoring all the relation with \( \delta \)-elements which is used to check both covering and strictness constraints. First, models that violate strictness constraints are checked with rules in (vii). Let us assume by contradiction that a dimension \( D^{\delta M} \) associated to a stable model of \( \Pi(D, \Sigma) \) violates a strictness constraint \( c_i \rightarrow c_j \). This implies that there exists elements \( a, b_1, b_2 \in E_D \) such that (i) \( \text{Cat}_D(a) = c_i, \text{Cat}_D(b_1) = c_i, \) and \( \text{Cat}_D(b_2) = c_j; \) and (ii) \( a \prec_{\delta M} b_1 \) and \( a \prec_{\delta M} b_2 \). By construction of \( D^{\delta M}, M \) contains \( R'(a, b_1, c_i, c_j) \) and \( R'(a, b_2, c_i, c_j) \). These atoms would violate rules in (vii). Thus, this is not possible and rules in (vii) discard all the candidate repairs in \( \text{CandRepair} \) that violate a strictness constraint.

Models that violate covering constraints are checked with rules in (viii). Let us assume by contradiction that a dimension \( D^{\delta M} \) associated to a stable model of \( \Pi(D, \Sigma) \) violates a covering constraint \( c_i \Rightarrow c_j \). This implies that there exists an element \( a \in E_D \) such that \( \text{Cat}_D(a) = c_i \) and there does not exist an element \( b \in E_D \) such that \( \text{Cat}_D(b) = c_j \) and \( a \prec_{\delta M} b \). By construction of \( D^{\delta M}, C(a) \in M \) and there exists no \( b \) such that \( R'(a, b, c_i, c_j) \in M \). Because of rules in (vi), there exists no \( b \) such that \( RT'(a, b, c_i, c_j) \in M \). Now, by rules in (viii) \( ax_{C_i}(a) \notin M \) and model \( M \) violates the denial constraint \( \leftarrow C_i(X), not\ ax_{C_i}(X), X \not\in \delta \). Thus, this is not possible and rules in (viii) discard all the candidate repairs in \( \text{CandRepair} \) that violate a covering constraint.

So far we have proved that the dimensions associated to the stable models of a program containing rules (i) to (viii) satisfy the constraints in \( \Sigma \). What we are missing now is to discard the models for which the associated dimensions are repairs which are not minimal. This is achieved through the weak constraints that were proven [14] to minimize the number of times atoms are made true. In this case, they will minimize the number of \( \text{Ins} \) and \( \text{Del} \) atoms in the stable models. This is, they will keep the optimal models among the stable models of the program.

**Completeness.** For a repair \( D' = (H_D, E_D, \text{Cat}_D, <_{\delta D}) \) of \( D = (H_D, E_D, \text{Cat}_D, <_{\delta D}) \) with respect to \( \Sigma \) we will construct a optimal model \( M' \) such that \( D^{M'} = D' \). Let \( M' \) be the union of the following sets:

\[
\begin{align*}
\{C(a)|a \in E_D \text{ and } C = \text{Cat}(a)\} \\
\{C(\text{del})|C \in H_0\} \\
\{R(a, b, c_i, c_j)|a <_{\delta D}, c_i = \text{Cat}(a) \text{ and } c_j = \text{Cat}(b)\}
\end{align*}
\]
\{R(a, \text{del}, c_i, c_j) | a \in E_D, c_i \not\rightarrow H_D c_j, \text{Cat}_D(a) = c_i, \text{ and } |\text{parent}(D, a, c_j)| = \emptyset\}

\{R'(a, b, c_i, c_j) | (a, b) \in <D', c_i = \text{Cat}(a) \text{ and } c_j = \text{Cat}(b)\}

\{R'(a, \text{del}, c_i, c_j) | a \in E_D, c_i \not\rightarrow H_D c_j, \text{Cat}_D(a) = c_i, |\text{parent}(D', a, c_j)| < |\text{parent}(D'_{\text{del}}, a, c_j)|\}

\{RT'(a, b, c_i, c_j) | (a, b) \in <D', c_i = \text{Cat}(a) \text{ and } c_j = \text{Cat}(b)\}

\{\text{Ins}(a, b) | (a, b) \in <D ' \text{ and } (a, b) \not\in <D\}

\{\text{Del}(a, b) | (a, b) \in <D ' \text{ and } (a, b) \not\in <D'\}

The dimension associated to \(M'\) is precisely \(D'\). Also, \(M'\) is a stable model of \(\Pi(D, \Sigma)\). Finally, since \(D'\) is a minimal repair, it is easy to check that \(M'\) is also a optimal model. \(\square\)