Efficient Intensional Redefinition of Aggregation Hierarchies in Multidimensional Databases

Mauricio Minuto Espil
Pontificia Universidad Católica Argentina
mminuto@uca.edu.ar

Alejandro A. Vaisman
Universidad de Buenos Aires
av2n@dc.uba.ar

ABSTRACT

Enhancing multidimensional database models with aggregation hierarchies allows viewing data at different levels of aggregation. Usually, hierarchy instances are represented by means of so-called rollup functions. Rollup between adjacent levels in the hierarchy are given extensionally, while rollups between connected non-adjacent levels are obtained by means of function composition. In many real-life cases, this model cannot capture accurately the meaning of common situations, particularly when exceptions arise. Exceptions may appear due to corporate policies, unreliable data or uncertainty, and their presence may turn the notion of rollup composition unsuitable for representing real relationships in the aggregation hierarchies. In this paper we present a language allowing augmenting traditional extensional rollup functions with intensional knowledge. We denote this language IRAH (Intensional Redefinition for Aggregation Hierarchies). Programs in IRAH consist of intensional rules, which can be regarded as patterns for: (a) overriding natural composition between rollup functions on adjacent levels in the concept hierarchy, (b) canceling the effect of rollup functions for specific values. Our proposal is presented as a stratified default theory. We show that a unique model for the underlying theory always exists, and can be computed in a bottom-up fashion. Finally, we present an algorithm that computes the revised dimension in polynomial time, although under more realistic assumptions, complexity becomes linear on the number of paths in the hierarchy of the dimension instance.

1. INTRODUCTION

The development of tools for OLAP (on-line analytical processing) has been calling the attention of the database community in the last six years. In models for OLAP [5, 15], data is represented as a set of *dimensions* and *fact tables*. Dimensions are usually organized as hierarchies, supporting different levels of data aggregation. Thus, facts can be viewed as points in a multidimensional space, with measures labeling these points. Usually, a dimension hierarchy is represented as a directed acyclic graph with a unique bottom level, and a unique distinguished top level, denoted *All*. Extensions for each aggregation hierarchy, called dimension instances, are represented as functions over level instances, called *rollup functions* [9, 10]. Only rollup functions between adjacent levels are provided, while rollups between non-adjacent levels are computed by means of function composition. Rollup functions allow defining how data is aggregated along a dimension hierarchy.

The model described above cannot express situations where exceptions occur [8]. For instance, in an insurance company, all costumers may be considered as "Reliable" if they are between forty and fifty years old, except those who have been fined for driving at high speed more than once. As another example, Mondays are usually considered as working days. Suppose, however, that a general failure in power supply arises some Monday, preventing people to work. That constitutes an exception. Exceptions also arise when rating financial investments, as we will show in the next section. In summary, when a hierarchy is built from inductive or imprecise knowledge, exceptions are likely to appear. We therefore introduce a language for maintenance of dimension hierarchies that allows defining exceptions that override the extension of some of the rollup functions implied in the hierarchies. The language aims at materializing exception paths not contemplated in the structure of dimensions, enhancing query processing while taking the exceptions into account.

1.1 Motivation

A credit company maintains a multidimensional database holding information about all of its loans, organized as follows: the loan identification code, the borrower identification, the identifier of the branch that approved the credit, the approval date, and the amount of the loan. The approved loans are:

loanId	borrowerId	Branched	date	amount
l_1	b_1	2	11/2000	15000
l_2	b_2	1	10/2000	3000
l_3	b_3	1	07/2000	250000
l_4	b_4	3	12/2000	13200

Here, *borrowerId*, *branchId* and *date* represent the bottom levels of dimensions *Borrower*, *Branch* and *Time*, respectively. *Amount* is a measure.

Figure 1(a) below depicts the schema of dimension *Borrower*, and Figure 1(b) below shows an instance of this dimension.

Each level in dimension Borrower is described by *attributes*.

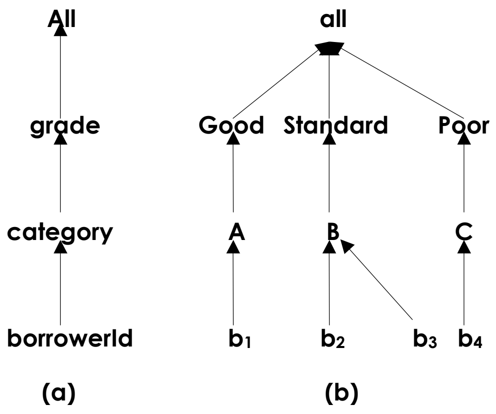

Figure 1: (a) Schema of dimension Borrower (b) An instance of dimension Borrower

Thus, for level *borrowerId*, attributes *name* and *income* are defined. Each borrower has a category in level *category*. Each category in level *category* corresponds to an income between two values represented by attributes *lower* and *upper*. Finally, level *grade* is described by attributes *lower* and *upper*, which define the bounds for interest rates corresponding to each grade. The following tables depict the values for the attributes described above, corresponding to the dimension instance of Figure 1(b).

borrowerId	name	income
b_1	J. Smith	90000
b_2	A. Logan	35000
b_3	P. Lew	25000
b_4	D. Egan	15000

category	upper	lower
A	unlimited	50000
B	50000	20000
C	20000	0

grade	upper	lower
Good	0.4	0.3
Standard	0.6	0.4
Poor	0.20	0.6

Example 1 *Let us suppose the following query: "List the total amount of loans summarized by grade". According to the hierarchy of Figure 1, borrowers b_2 and b_3 will contribute to category "B", which receives a total of U$S 253,000, if we consider the fact table above, and category "B" will contribute to grade "Standard", which receives the same amount. However, assume that although the income of customer b_3 determines that she belongs to category "B" (i.e. her loans will contribute to grade "Standard"), we are interested in giving her loans a better grade, say "Good". Thus, we define the following exception: "borrower b_3 must be graded "Good". In this case, grade Good" will totalize U$S 265,000, while grade "Standard" will totalize only U$S 3,000.*

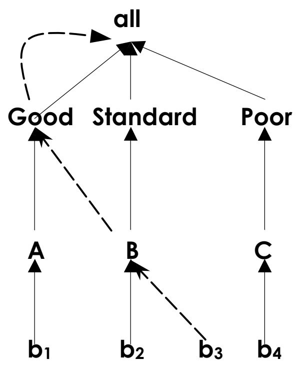

Figure 2: Modified instance of dimension Borrower

In order to tackle situations like the one presented in Example 1, we must modify the instance in Figure 1(b). The new extension of dimension *Borrower* is shown in Figure 2. The dashed lines indicate the exception path that will be followed by the aggregation algorithm when computing the result of the query.

1.2 Contribution

In this paper we present a language that allows expressing intentional redefinitions of aggregation hierarchies like the one in Figure 1. We denote this language IRAH (standing for Intensional Redefinition of Aggregation Hierarchies). IRAH also provides a way of handling contradictory exception expressions. Formally, we regard dimension instances and statements in IRAH programs as embedded in a default reasoning framework, in order to give the language a precise semantics that entails the existence of a unique preferred model for the constrained dimension instances.

As exposed above, IRAH is intended mainly as a maintenance language. When exceptions occur, an IRAH program should be defined and executed, thus producing a revision on the paths in the dimension instance. Hence, we present an algorithm that computes the revised paths. We show that the time and space complexities of this algorithm lie within PTIME and PSPACE, respectively.

1.3 Paper Outline

The remainder of this paper is organized as follows: In Section 2 we present the model and the IRAH language. In Section 3 we give the language's semantics. In Section 4 we present an algorithm for revising extensions of dimension hierarchies, a comprehensive example, and outline the algorithm's complexities. In Section 5 we compare our approach with related work. We conclude in Section 6 and propose future lines of research.

2. THE MODEL

Although we represent dimensions following the ideas of Hurtado et al. in [9, 10], we must adapt their model in order to apply revisions to dimension instances. Basically, we relax some of the constraints imposed there.

2.1 Dimension Hierarchies

Let us consider the following finite sets: a set **D** of dimension names, a set **L** of level names, a set **C** of constants, and a set **P** of abstract paths.

Definition 1 (Dimension Schema): *A dimension schema is a triple (d, L, $\preccurlyeq_d$), where d is a name in* **D**, *L is a set included in* **L**, *and $\preccurlyeq_d$ is a relation in L × L such that $\preccurlyeq_d$*, its transitive closure, is a partial order, with a unique bottom, denoted* bottom, *and a unique top, denoted* All, *such that* bottom ≠ All. *An element l in L is called a level of dimension d.*

Definition 2 (Instance Sets of Levels): *Let ρ be a relation in* **D** × **L** × **C**. *Relation ρ associate constants (from now on coordinates) with levels of a dimension; a tuple (d, l, c) ∈ ρ means that c is a coordinate in level l of dimension d. The set Iset (d:l)* = { c │ c ∈ **C**, (d,l,c) ∈ ρ } *is called the instance set of level l of dimension d. Moreover, there exists a constant* all ∈ **C** *such that Iset (d:*All) = { all }, ∀*d* ∈ **D**.

Definition 3 (Dimension Instance): *A dimension instance of dimension d is a relation, call it* path_d, *in* **D** × **P** × **L** × **C**, *with d as the first column value of all rows. A tuple (d, p, l, c) ∈* path_d *means that an abstract path p has c as coordinate for level l of dimension d, and implies that the tuple (d, l, c) ∈ ρ. No two tuples in relation* path_d *with the same level name and the same abstract path have different coordinates in the last column.*

Definition 4 (Paths): *Given a dimension d, an abstract path p in an instance* path_d, *induces a sequence, a (concrete) path* $< l_1$: c_1,..., l_k: $c_k >$, *such that for all positions j and j+1 in the sequence,* 1≤*j*≤k-1, $c_j \in$ *Iset* (*d*:l_j) *and* $c_{j+1} \in$ *Iset* (*d*:l_{j+1}), *and* $l_j \preccurlyeq_d l_{j+1}$ *holds.*

2.2 The IRAH Language

Dimension instances are usually presented in the form of rollups, which are interpreted as mappings between level instances. In order to tackle the redefinition problem, that is, the possibility of overriding predefined rollup functions, we provide a language which allows expressing this overriding as a set of rules such that a priority can be defined between them. We call this language IRAH (Intensional Redefinition of Aggregation Hierarchies). We first present the language's syntax; in the next sections we will show how to map predefined rollups and IRAH rules to normal default schemas and give an interpretation to these defaults such that a unique model exists for them.

IRAH is a typed language; variables and constants are typed. Types in IRAH are: **D** for dimension type; **L** for level type; **C** for coordinate type. Usual basic types as integers, booleans, characters, strings, dates, and so on, are also supported. A special symbol '=' denotes the equality predicate for each type. Analogously, the usual symbols '≤', and '≥' are given the usual meaning. A set of strongly typed function signatures is included in the language; and an implicit set of functions **Desc** (standing for level descriptors) is assumed, in order to admit expressions composed of references to level attributes. Variables in IRAH must be declared.

The basic construct in IRAH is a *coordinate expression.* A *coordinate expression* is an expression of the form d:l:t, where d is a constant of type **D**, l is a constant of type **L**, and t is a variable or a constant (a term) of type **C**. When a dimension d is implicit the prefix d: may be omitted. Informally, a coordinate expression asserts that a path with term t in level l exists in the instance of dimension d.

Notation. In the examples that follow, for the sake of brevity, we will represent variables with the letters X, Y, and Z. Recall, however, that variables in IRAH must be declared.

Example 2: *Assuming Borrower as the implicit dimension, the following are coordinate expressions corresponding to Figure 1.*

Borrower:category:B; Borrower:borrowerId:b_3; grade:Standard;

A *level expression* is an expression of the form φ(*v*.A, t_1,..., t_m), where φ is a predicate symbol (a boolean function), *v* is a variable of type **C**, A is a descriptor of some level *l*, and t_1,..., t_m m ≥ 0, are well formed ground terms of types T_1,..., T_m respectively, matching the signature of φ. Level expressions define subsets of the instance set of level *l.*

Example 3: *The expression* X.income=1000*, is a level expression stating that income in the* borrowerId *level equals 1000. Note that* X *is a variable.*

We can build *level formulae* from level expressions and propositional connectors ∧,∨ and ¬ in the usual manner. For instance, the construct that follows is a level formula:

X.income ≥ 1000 ∧ X.income ≤ 1500.

A *coordinate formula* is defined as follows: (a) every coordinate expression is a coordinate formula; (b) a conjunction of a non-ground coordinate expression ξ containing a level variable L, and a level formula bound to ξ by L, is a coordinate formula.

Example 4: Examples of coordinate formulae are:

Borrower:category:B; Borrower:borrowerId:X ∧ X.income ≤ 1500.

A *redefinition rule for dimension d* is an expression of the form: $B_1(l_1)$,...,$B(l_k)$ / l:c, where l_1,...,l_k, l are levels of *d*, $l_j \preccurlyeq_d^* l_{j+1}$, 1 ≤ j ≤ k-1, $l_j \preccurlyeq_d^*$ l; and $B_1(l_1)$,...,$B(l_k)$ are coordinate (*body*) formulae for levels l_1,...,l_k, respectively, c is a coordinate in level l. We call the coordinate expression l:c the *head* of the rule.

Example 5: *The exception in Example 1 is expressed in IRAH as:* Borrower:borrowerId:b_3 / grade:Good. *The intuitive meaning of the rule is that every path of the form* %borrowerId:b_3%, *belonging to an instance of dimension Borrower, should match the path:*

% borrowerId:b_3%grade:Good%

with % a wild card substitutable for any subpath of length ≥ 0..

Example 6: *Let us present another example of an exception: "Money lent to a borrower with an income under U$S 28,000 belonging to a category with a lower bound over U$S 18,000, should be graded Poor" This exception acts as a constraint, reducing risks by means of reclassifying borrowers with categories that admit lower bounds for income. In our running example, borrower* b_3 *will be affected. The exception in IRAH reads:*

(borrowerId:X $\wedge$ X.income < 28000), (category:Y $\wedge$ Y.lower > 18000) / grade:Poor. *This rule states informally that paths of the form*: %borrowerId:X% *that match paths of the form*: %category:Y%, *for borrowers* X *with income underU$S 28,000 and categories* Y *with lower bound over U$S 18,000, should match paths* %grade:Poor%.

We proceed, in what follows, to give precise formal semantics to IRAH programs. The semantics presented therein are not intended for implementation purposes, but for capturing accurately the meaning of the revision process implied by the programs. Implementation issues will be discussed in Section 4.

2.3 Default Reasoning

Our approach is based formally on the concept of extensions of a prioritized default theory [1, 2, 4]. Thus, we briefly review this concept, and refer the reader to the bibliography for details.

Let α and γ be first order sentences (formulae with no free variables). An expression δ of the form:

$$\frac{\alpha}{\gamma}$$

is called a *normal default rule* [16] (in what follows, simply a *default*). α is called the *prerequisite* of δ, denoted pre(δ), and γ represents simultaneously the *justifications* and the *consequent* of δ, denoted just(δ) and cons(δ), respectively. The following statement can express the intuitive meaning of a default δ: *"if we believe in* α*, and* γ *does not contradict our beliefs, we can infer* γ*"*. If we allow α and γ to have free variables, we have *default schemas*, representing all possible default instances built upon substitution. A default rule δ is said to be *applicable* to a set E of beliefs (first order sentences), closed under implication, if and only if pre(δ)$\in$ E and $\neg$just(δ)$\notin$ E. Consider a consistent set of beliefs W, a set Δ of defaults. A pair T = (W, Δ) is called a normal default theory. If we add a partial order $\prec$ on defaults in Δ., we call theory T a *prioritized default theory* w.r.t. $\prec$. The meaning of a prioritized default theory is given operationally, by means of so-called *prioritized extensions*, as follows: Let π be a sequence of defaults in Δ, such that: $\pi[1 - 0]$ is the null sequence; $\pi[i]$ is the default that occurs in position i of π; $\pi[1- j]$ is the prefix of length j of π; $\pi[\ k -\ ]$ is the suffix of π, starting from position k. A sequence π is said to be a *process* for the theory T, if and only if the following condition holds: for every position k in π, the default $\pi[k]$ is applicable to a set of beliefs $In(\pi[1 - (k\text{-}1)])$, defined as the classical theory for the axiom set W $\cup$ {cons(δ) | δ occurs in $\pi[1 - (k\text{-}1]$}.

A process π is said to be *generated by* a strict well order $\ll$ if and only if, for all positions i in π, $\pi[i]$ is the minimal default w.r.t $\ll$that occurs in $\pi[i -]$ and is applicable to $In(\pi[1 - (i\text{-}1)])$.

Finally, a set of beliefs E is said to be a *prioritized extension* of a theory T= (W, Δ) w.r.t. a partial order $\prec$, if and only if there exists a strict well order $\ll$ on defaults in Δ which contains $\prec$, and E = $In(\pi)$ for some process π generated by $\ll$.

2.4 Rollups and IRAH Rules as Defaults

Let us have a rollup between coordinates *a* and *b* in levels l_1 and l, in dimension instance *d*. We can map this rollup to a default schema of the form:

$$\frac{path_d\,(d, \mathrm{X}, l_1, a)}{path_d\,(d, \mathrm{X}, l, b) \wedge \mu_b}$$

where $\mu_b = \bigwedge_{v\in \mathbf{I},\ v\neq b} \neg path_d(d, \mathrm{X}, l, v)$, $\mathbf{I} = Iset\,(d : l)$.

We call a default schema like the above an *l*-default schema. The expression μ_b is a finite conjunction of negative literals of the form $\neg path_d(d, \mathrm{X}, l, v)$, for each coordinate value $v \neq b$ in set *Iset*(*d:l*), and is called a *uniqueness guarantee* for coordinate *b* in level *l* of dimension *d.* A uniqueness guarantee ensures that a path in a dimension instance has no more than one coordinate defined for each level.

Example 7: *Let us analyze Figure 1(b) again*: *The rollup between coordinate* B *in level* category *and coordinate* Standard *in level* grade, *can be mapped to the following normal default schema:*

$$\frac{path_{Borrower}\,(\text{Borrower, X, category, B})}{path_{Borrower}\,(\text{Borrower, X, grade, Standard}) \wedge \mu_{\text{Standard}}}$$

$$\text{where } \mu_{\text{Standard}} = \neg path_{Borrower}(\text{Borrower, X, grade, Good}) \wedge.$$

$$\neg path_{Borrower}(\text{Borrower, X, grade, Poor})$$

Analogously, IRAH rules can be mapped to default schemas. First, we can map coordinate expressions of the form *d: l: c* appearing in the rule, to atoms of the form $path_d\,(d, \mathrm{X}, l, c)$. The prerequisite of the default schema is then built out of the conjunction of all transformed formulae in the body of the rule; and the justification-consequent of the default schema is built out of the conjunction of: (a)- the transformed formula in the head of the rule and (b)- the uniqueness guarantee for its associated coordinate. A complete description of the mapping process could be found in [14].

Example 8: *The rule in example 5 can be mapped to the default schema:*

$$\frac{path\,(\text{Borrower, X, borrowerId}, b_3)}{path\,(\text{Borrower,X,grade,Good}) \wedge \mu_{\textbf{Good}}}$$

$$\text{where } \mu_{\textbf{Good}} = \neg path_{Borrower}(\text{Borrower, X, Grade, Standard}) \wedge$$

$$\neg path_{Borrower}(\text{Borrower, X, Grade, Poor})$$

We can map each rollup in a dimension instance and each rule in an IRAH program to normal default schemas. We can build a set Δ with these schemas.

2.5 Priorities and Contradictory Exceptions

The following definition yields a *partial* order $\prec$ between defaults in Δ.

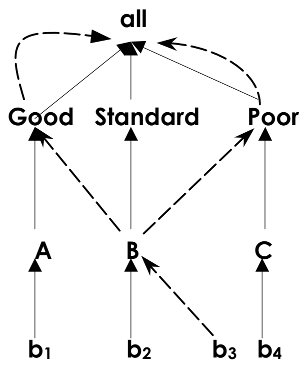

Figure 3: Contradictory exceptions over dimension Borrower

Definition 5: **(Priority Constraint)** ***According to*** *$\prec$, every instance of an l-default schema in set Δ, resulting from the mapping of an IRAH rule must precede all instances of l-defaults resulting from the mapping of rollups, for each level l.*

It could be the case that contradictory exception rules appear within an IRAH program. For instance, let us assume two exceptions holding over dimension *Borrower*: the exception in Example 1, and the following one: "A borrower with income between U$S 20,000 and U$S 30,000 should be graded *Poor*". In this case, a path departing from b_3 matches this exception, but also matches the previous one. The situation is shown in Figure 3 above, where dashed lines correspond to alternatives for the path departing from b_3.

The former situation may lead to assign different grades to borrower b_3: Good and Poor. These assignments contradict uniqueness guarantees. Two approaches could be followed: a *credulous* approach or a *skeptical* approach. A credulous approach leads to accept both alternatives, yielding alternative hierarchies, an undesired result. A skeptical approach, on the opposite, prevents grading loans from borrower b_3. We have chosen the second approach. Thus, the path departing from b_3 is now undecided in level *grade*.

3. SEMANTICS

In the previous section we showed how each member of a rollup function and each IRAH rule can be mapped to a normal default rule schema, building a set Δ of these rule schemas. In this section we provide axioms and build a theory on them.

Due to uniqueness guarantees, a path in an instance of a dimension d is uniquely determined by a coordinate in l_{bottom}, the bottom level of the dimension hierarchy. We may assume therefore that a one-to-one mapping η between coordinates in the bottom level and abstract paths in **P** exists, and build an axiom set Γ of ground atoms of the form $path_d(d,\eta(b),l_{bottom},b)$, for each coordinate b in the instance set of level l_{bottom}. We also need to state that the coordinate *all* is the coordinate value in level *All*, for all paths. Thus, we need to augment Γ with atoms of the form: $path_d(d,p,All,all)$, for each path p in the range of η. For instance, in our running example, ground atoms:

$$path_{Borrower}(Borrower,\ \eta(b_3),\ borrowerId,\ b_3) \text{ and}$$

$$path_{Borrower}(Borrower,\ \eta(b_3),\ All,\ all)$$

would be axioms in Γ.

Now we can provide semantics to dimension instances under redefinition. Let Γ be the set defined above, and Δ and $\prec$ be the sets of defaults schemas and the partial order defined in Section 2.

Definition 6: *A model for a dimension instance given by a set of rollups and a set of IRAH rules, is the Herbrand interpretation of a set of formulae ξ such that $\xi = \xi_i$ with i maximal, where sets ξ_i are inductively defined as follows:*

- $\xi_0 = \Gamma$
- $\xi_i = \bigcap \text{Pos}\ (\ \{\ \varepsilon\ |\ \varepsilon \text{ is a prioritized extension of}$

$$(\xi_{i-1}, \Delta_i) \text{ w.r.t } \prec \}\ )$$

where Pos stands for the positive literal fragment of the set of formulae given as argument, and Δ_i stands for the subset of Δ containing l-default rule schemas, such that l=θ(i), with θ defined as a mapping from levels into consecutive non-negative integers, satisfying: (a)- θ(Bottom)=0; (b)- *for every pair of levels l_1 and l_2, such that $l_1 \preccurlyeq_d l_2$, $\theta(l_1)<\theta(l_2)$ holds. We call θ a stratum mapping.*

Clearly, the semantics above interpret roll-ups and redefinition rules in the relation $path_d$. Note also that the chosen semantics imply that, if two conflicting defaults become applicable under $\prec$, no tuple would be present in $path_d$ as a conclusion. Thus, we may have *incomplete paths*. Incomplete paths are useful when modeling uncertainty and the chosen approach is *skeptical*.

Lemma 1 *There exists at least one prioritized extension ξ of each theory* (ξ_{i-1}, Δ_i) w.r.t. $\prec$*, with ξ_{i-1} and Δ_i defined as above.*

Lemma 2 *The positive literal fragment of a prioritized extension ξ of each theory* (ξ_{i-1}, Δ_i) w.r.t. $\prec$ *is finite.*

The proofs for these lemmas can by found in [14].

4. AN ALGORITHM FOR REVISION

The semantics presented in Section 3, interprets dimension instances as tuples in the relation *path.* This relation, however, is clearly not a good choice for representing a dimension instance, in computational terms. We can do better representing paths by means of tables. For instance, for the revised dimension of Figure 2 we would get the following dimension table:

borrowerID	category	grade	All
b_1	A	Good	all
b_2	B	Standard	all
b_3	B	Good	all
b_4	C	Poor	all

Algorithm 1 below takes a table d_{inst} representing a dimension instance, and a set of IRAH rules as input, and builds the revised dimension instance as output (modified paths only). The algorithm visits IRAH rules and minimizes the number of visits to paths in the dimension instance.

We represent levels and identify IRAH rules by means of consecutive non-negative integers. We define the following data structures: (a) *Cond*, a two-dimensional sparse array on rules and levels. Each nonempty cell Cond[i,j] points to the coordinate formula for level j (body or head) of rule i; (b) an array *Srules* over levels. Each cell Srules[j] points to the set of rules with the first body formula in level j; (c) an array *Erules* over levels. Each cell Erules[j] points to the set of rules with level j in the head; (d) a set *Paths* of candidate paths (tuples with levels as attributes). A Boolean variable for each level of a path in Paths is provided, indicating whether the coordinate for the level has been modified or not; (e) a set *TargetPaths* of objective paths; (f) an array *Rpaths* over rules. Each cell Rpaths[i] points to the set of paths in Paths satisfying the formula in the body of rule i; (g) an array *Prules* over paths. Each cell Prules[k] points to the set of rules i such that path k satisfies the formula in the body of rule i.

Algorithm 1 (Revision)

```
Main
Build data structures from rules;
FOR levels j from 1 to maxlevels DO
    Revise_Paths (j);
    Activate_Paths (j);
    Delete_Paths (j);
END FOR
output set Paths;

Revise_Paths (level j)
A = {};
FOR EACH rule i in Erules[j] DO
    A = A ∪ { coordinate c in Cond[i,j] };
    TargetPaths = { paths p in dimension d | p.j in A };
/* p is the subpath including the levels above level j in the hierarchy.
    FOR EACH rule i in Erules[j] DO
        FOR EACH path k in Rpaths[i] DO
            IF not Test_Conflict (i, k)
                Merge (j, k, path p in TargetPaths | coordinate c in
                Cond[i, j] );
            ELSE Put_nulls (k, j),
            END IF
        END FOR
    END FOR
END FOR

Merge (level j, path k, path p)
Set coordinate c in level j of path p, as the new coordinate for
level j of path k;
FOR EACH level l of path k s.t. j $\preccurlyeq_d$* l holds DO
    Set coordinate c in level l of path p as the new coordinate for
    level j of path k, provided the coordinate has not been yet
    modified; otherwise put null as coordinate for level j of path
    k.
END FOR

Put_nulls (path k, level j)
Put null in j and in every level l of path k s.t. j $\preccurlyeq_d$* l holds.

Test_Conflict (rule i, path k ) /* returns a Boolean
IF (i.head.level=r.head.level and
    i.head.coordinate ≠ r.head.coordinate)
    for some rule r in Prules[k]
    return (true);
ELSE return (false);
END IF

Activate_Paths (level j)
FOR EACH rule i in Srules[j] DO
    A={path p in Paths | Cond[i,j] holds in p};
    FOR EACH path k in A DO
        add k to Rpaths[i]; add i to Prules[k];
    END FOR
    A={path p in table $d_{inst}$ | Cond[i,j] holds in p};
    FOR EACH path k in A not already in Paths DO
        add k to Paths; add k to Rpaths[i]; add i to Prules[k];
    END FOR
END FOR

Delete_Paths (level j)
FOR EACH rule i = 1 to maxrules
    such that Cond[i, j] is nonempty DO
    IF (i not in Srules[j] and i not in Erules[j]
        A={path p in Rpaths[i] | Cond[i, j] does not hold in p };
        FOR EACH path k in A DO
            delete k from Rpaths[i]; delete i from Prules[k];
            IF Prules[k] is empty
                delete k from Paths;
            END IF
        END FOR
    END IF
END FOR
```

Algorithm 1 visits the hierarchy of levels in a bottom-up manner. First, head formulae in the current level are examined, and a revision is performed on the current level in non-conflicting paths. Every body formula involved in a level is then examined. If the formula is the first body in a rule, and a path in memory satisfies the condition, the path is added to the set of paths potentially involved in the rule. Paths satisfying the condition, and not present in memory, are retrieved from the dimension instance derived from rollups. If the formula is not the first formula in the body of a rule, we unlink from the rule every path not satisfying the condition. Finally, the algorithm outputs the revised paths in memory.

Example 9: *Let us consider a dimension* Employee*, with levels* emp, unit, group *and* division*, such that* emp $\preccurlyeq$ unit*,* unit $\preccurlyeq$ group*,* group $\preccurlyeq$ division*,* division $\preccurlyeq$ All. *Throughout the example we will use, we will be using the following table describing level* unit:

rowID	unit	unit name
u_1	U_1	Software.
u_2	U_2	Comm.
u_3	U_3	Med. Equip.
u_4	U_4	Economics
u_5	U_5	Engineering

We assume that u_i *is the row identifier, and* U_i *is the corresponding value in the instance set of the level* unit*. Column* unit_name *is a level attribute. Analogously, tables describing levels* group *and* emp *have values* g_i, G_i *and* e_i, E_i *in columns* rowID, group *and* rowId, emp, *respectively. We do not show the rest of the attributes of those tables because they are irrelevant to the case example*

The following table represents an instance of dimension Employee. *Column* pathId *is the row identifier.*

pathID	empID	unitID	groupID	DivisionID
p_1	e_1	u_1	g_1	d_1
p_2	e_2	u_1	g_1	d_1
p_3	e_3	u_1	g_1	d_1
p_4	e_4	u_2	g_1	d_1
p_5	e_5	u_2	g_1	d_1
p_6	e_6	u_3	g_2	d_1
p_7	e_7	u_3	g_2	d_1
p_8	e_8	u_4	g_3	d_2
p_9	e_9	u_5	g_4	d_3
p_{10}	e_{10}	u_5	g_4	d_3

The following IRAH program represents exception rules. Note that conditions are not stated over row identifiers, but over level attributes:

1. emp: W $\wedge$ (W.emp = E_1 $\vee$ W.emp = E_4), unit: Y $\wedge$ Y.name = 'Software' / group : G_2
2. emp : E_{10} / group : G_3
3. group : G_3 / division : D_1
4. unit: W $\wedge$ W.unit_name = 'Comm.' / group : G_2
5. emp: E_4, unit: Y $\wedge$ Y.unit_name = 'Comm.' / group: G_3
6. unit: W $\wedge$ W.unit_name = 'Comm.' / division : D_1

Initially, the two dimensional array Cond *is filled as follows*:

rule #	level emp	level unit	level group	level division
1	emp = E_1 $\vee$ emp = E_4	unit_name ='Software'	G_2	
2	E_{10}		G_3	
3			G_3	D_1
4		unit_name = 'Comm.'	G_2	
5	E_4	unit_name = 'Comm.'	G_3	
6		unit_name = 'Comm.'		D_2

Arrays Srules and Erules are*:*

Srules (level emp) = { rule 1, rule 2, rule 5 }; Srules (level unit) = { rule 4, rule 6 }; Srules (level group) = { rule 3 }; Srules (level division) = { };

Erules (level emp) = {}; Erules (level unit) = {}; Erules (level group) = { rule 1, rule 2, rule 4, rule 5 }; Erules (level division) = { rule 3, rule 6 };

The sets Paths, Rpaths *and* Prules *are all empty*.

In the first iteration (for level emp*), since the set Erules for level* emp *is empty, procedure Revise_Paths does nothing; Activate_Paths, adds new rows to the set* Paths*, yielding:*

pathID	empID	unitID	groupID	DivisionID
p_1	e_1	u_1	g_1	d_1
p_4	e_4	u_2	g_1	d_1
p_{10}	e_{10}	u_5	g_4	d_3

Rpaths *and* Prules *are updated as follows*:

Rpaths(rule 1) = { p_1, p_4, p_5 }, Rpaths(rule 2) = { p_{10} }, Rpaths(rule 5) = { p_4 }

Prules(path p_1) = { rule 1 }, Prules (path p_4) = { rule 1, rule 5 }, Prules (path p_5) = { rule 1 },

Prules (path p_{10}) = { rule 2 }

Function Delete_Paths does nothing because there is no rule not present in Srules (level emp) *nor in* Erules (level emp) *such that Cond is nonempty in column* emp.

The algorithm proceeds analogously for the other levels. We omit describe the iterations due to space limitations. The complete example could be found in [14].

The final state of set Paths*, that is, the output of the algorithm, is shown in the table below:*

PathID	empID	unitID	groupID	DivisionID
p_1	e_1	u_1	g_2 (g_1)	d_1 (d_1)
p_4	e_4	u_2	null (g_1)	d_2 (null, d_1)
p_8	e_8	u_4	g_3	d_1 (d_2)
p_{10}	e_{10}	u_5	g_3 (g_4)	d_1 (d_2, d_3)

Old coordinate values for each level in the paths are shown enclosed in parentheses. Note, for instance, that path p_4 has null as coordinate for level group*. This is because a conflict between rules 4 and 5 arises. Path p_{10} has been updated twice in level* division: *the first time according to rule 2 (when updating level* group*); the second time according to rule 3. A similar situation occurs with path p_4. An overriding of a previously assigned null value has occurred in this case.*

Theorem 1 *Algorithm 1 belongs to PTIME and PSPACE complexity classes. Moreover, if we assume that the number of paths satisfying the conditions in the bodies of rules is bound, the algorithm indeed is linear on the size of the dimension instance.*

Proof. See [14].

5. RELATED WORK

Several techniques have been proposed for dealing with irregular hierarchies, as many-to-many dimensions [12], and multiple hierarchies [15]. The former technique assigns probabilistic weights to roll-up alternatives, yielding probabilistic aggregations. The latter deals with alternatives in the lattice. However, the approach is strictly operational, and no priorities are imposed among alternatives. Studies of irregular hierarchies can be found in works on hypothetical queries for OLAP [3], and on multiple scenarios [13]. While these works deal with "what if" sort of queries, and do not consider rollups as functions, our work incorporate exceptions from the beginning with the concept of rollup function embedded.

Previous work has studied dimension updates. A set of basic and complex operators has been proposed [9, 10] in order to update dimension hierarchies. These operators, however, cannot capture exceptions. Incomplete hierarchies have also been explored by Kimbal et al. in [12]. Their solution adds "not known" values to level instances. If exceptions occur, these values must be propagated up in the hierarchy. No clear semantics is provided, however, for null values. To the best of our knowledge, our work is the first one explicitly dealing with exceptions that override composition of rollup functions.

With respect to the semantics of IRAH, we can find a close resemblance between our approach, and the notion of stratified default logic extension [6]. Our model may be viewed, under this notion, as an extension of a stratified default theory, skeptical at the level of strata.

As a final comment, the choice of normal defaults representing rules in IRAH could be easily modified choosing semi-normal default rules [7]. We must simply omit uniqueness guarantees in consequents, in order to avoid loss in meaning. However, if we want to extend the semantics, and consider three-value models, our semantics applied to semi-normal theories fails to provide the same meaning.

6. CONCLUSION

We have introduced IRAH, a language of intensional rules allowing redefining dimension instances, in order to support exceptions that partially override roll-up composition, and cancel the effect of rollups in the presence of contradiction. We have presented clear semantics for rollups and IRAH rules together, based on a prioritized stratifiable default theory; a model for the underlying theory has been precisely defined. We have also introduced a polynomial time and space algorithm that computes the revised paths in the dimension instance, according to IRAH rules.

A clear future step consists in developing efficient algorithms for revising materialized views with aggregation, particularly cube views. In this sense, the algorithm presented in this paper clearly identifies the coordinate changes produced by the revision process in the aggregation paths.

Our approach to the semantics of the entire model can be slightly changed, to include uncertain and negative knowledge (three-value models), admitting the use of negation in the body and the heads of rules, thus augmenting the expressive power of the language. These changes yield an interesting question: how these kinds of rules, different from those presented in this paper, can be exploited in modeling multidimensional data with constraints. Exploiting priorities for modeling plausibility on inconsistent data sources is another interesting research topic arising from the work presented here. We are currently working on using non-monotonic reasoning for expressing aggregation.

7. REFERENCES

[1] Antoniou, G. A tutorial on default logics. ACM Cpmputing Surveys, :Vol. 31. No. 4 , 1999.

[2] Antoniou, G.: Nonmonotonic Reasoning. The MIT Press, Cambridge, Massachusets, 1997.

[3] Balmin, A.; Papakonstantinou, Y.; Papadimitriou, J.: Optimization of Hypothetical Queries in OLAP environments. Proceedings of IEEE/ICDE'99, 1999.

[4] Brewka, G.: Reasoning about Priorities in Default Logic. In Proceedings of the 12th National Conference on Artificial Intelligence, 1994.

[5] Cabibbo, L.; Torlone, R.: A Logical Approach to Multidimensional Databases. In EDBT'98: 6th International Conference on Extending Database Technology, Valencia, Spain. 1998.

[6] Cholewinski, P.: Stratified default Logic. Proceedings of the Intl. Conf. on Logic Programming, 1994.

[7] Etherington, D.: Formalizing Non-monotonic Reasoning Systems. Artificial Intelligence 31, 1. 1987

[8] Etherington, D.; Reiter R.: On Inheritance Hierarchies with Exceptions. Proceedings of the AAAI, 1983.

[9] Hurtado, A.;Mendelzon, A.;Vaisman, A.: Maintaining data cubes under dimension updates. Proc. IEEE/ICDE, 1999.

[10] Hurtado, A.; Mendelzon, A.; Vaisman, A.: Updating OLAP dimensions. Proceedings of ACM DOLAP, 1999.

[11] Kimball, R.: The Data Warehouse Toolkit. John Wiley & Sons. 1996.

[12] Kimball, R.; Reeves, L.; Ross, M.; Thornthwaite W.: The Data Ware- house Lifecycle Toolkit. John Wiley & Sons, 1998.

[13] Lehner, W.: Modeling Large Scale OLAP Scenarios. 6th Int'l Conf. EDBT'98. 1998.

[14] Minuto Espil, M.; Vaisman, A.: Efficient Intentional Redefinition of Aggregation Hierarchies in Multidimensional Databases. Full paper. 2001. Available at URL address: http://www.cs.toronto.edu/~avaisman/publications.

[15] Pourabbas, E; Rafanelli, M.: Characterization of Hierarchies and Some Operators in OLAP Environments. Proceedings of ACM DOLAP, 1999.

[16] Reiter, R.: A logic for default reasoning. Artificial Intelligence 13, 1980.