A GRAMMAR-BASED APPROACH TOWARDS UNIFYING HIERARCHICAL DATA MODELS (extended abstract)

Marc Gyssens, Jan Paredaens, Univ. of Antwerp (UIA), B-2610 Antwerpen, Belgium
Dirk Van Gucht, Indiana Univ., Bloomington, IN 47405-4101, USA

A simple model for representing the hierarchical structure of information is proposed. This model, called the grammatical model, is based on trees that are generated by grammars; the grammars describe the hierarchy of the information represented by the trees. Two transformation languages, an algebra and a calculus, are presented and shown to be equally expressive.

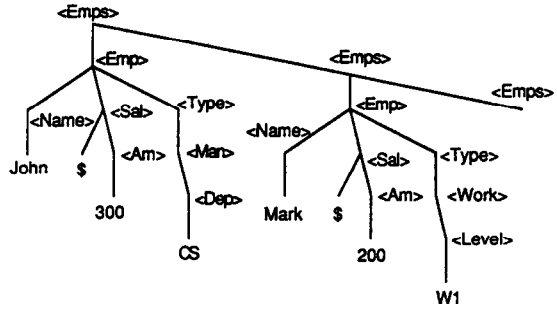
1. Introduction

Until the mid-eighties a lot of attention was paid to the relational database model (see e.g. [19,20,24]). We were intrigued by its simplicity, both to model and manipulate data. Recently, however, we became aware of its drawbacks when trying to model non-traditional data applications, such as CAD-CAM, office automation, text-oriented and multimedia databases. Therefore, several alternatives have been proposed.

Semantic data models, such as ER [6], FDM [22], SDM [13], Format [14] and IFO [2] provide a rich set of design tools for representing complex interrelationships of data not present in the relational model, such as aggregation, generalization and set constructs. Logic based models, such as Datalog [24], LDM [18] and LDL [4] zero in on the limited expressiveness of data manipulation languages of the relational model, i.e. they generalize the relational calculus to express recursively specified queries. Finally, relational extensions of the standard relational model, such as RT/M [7] and the nested model [9,15,23], try to strike a balance between generalizing the data modeling and the data manipulation parts of the relational model.

All these models share the property that they recognize the most fundamental characteristic of data to be its hierarchical structure. On the other hand, however, it is not quite clear whether they can effectively model all data applications which exhibit an hierarchical nature. A good example are textbases [11], which in addition to having an hierarchical structure, are constructed out of rules that follow a grammatical structure. Grammatical structures are also implicitly present in e.g. Verso [1,5], a variation of the nested relational model, where at some level, data are structured as regular expressions.

Recently, several authors (e.g.[11,17,21]) used the well-understood concept of grammars for formal languages [10] to model data. In this paper, we present the grammatical model as a unifying skeleton to describe the hierarchy in an information base as well as the structure of texts. This model has been presented informally in [11]. It represents the hierarchical structure of information as a tree which can be generated by a grammar. In this way, the information about two employees (one manager and one worker) will be represented by the tree:



This tree is generated by a grammar with productions:

P1 : $\langle Emps \rangle \rightarrow \langle Emp \rangle \langle Emps \rangle$
P2 : $\langle Emp \rangle \rightarrow \langle Name \rangle \langle Sal \rangle \langle Type \rangle$
P3 : $\langle Type \rangle \rightarrow \langle Man \rangle$
P4 : $\langle Type \rangle \rightarrow \langle Work \rangle$
P5 : $\langle Man \rangle \rightarrow \langle Dep \rangle$
P6 : $\langle Work \rangle \rightarrow \langle Level \rangle$
P7 : $\langle Sal \rangle \rightarrow \$\langle Am \rangle$

Notice that production P1 specifies information as a set. Production P2 specifies an employee as the aggregation of his/her name, salary and type. P3 and P4 define the type of an employee as either manager or worker, which is an example of generalization. Hence the grammatical model allows for the three basic constructs of most data models in a uniform way [8,21]. Finally notice production P7. The grammatical model allows, in a convenient way, for additional syntactic features, e.g. the terminal symbol "$" indicates that the salary is expressed in dollar amounts.

There are two obvious ways to express queries by transforming the trees. A first method consists in defining operators that locally transform the trees and the grammars. A second way consists in describing this

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.
© 1989 ACM 0-89791-317-5/89/0005/0263 $1.50

transformation in a less procedural way, indicating the relationship between the given trees and the result trees. For reasons that are evident these methods are called the algebra and the calculus, respectively. They are shown to be equivalent. Before describing these two query languages, however, we first give the basic definitions of the grammatical model.

2. The Grammatical Model

Throughout this paper, we assume the reader is familiar with the basic terminology concerning trees (e.g. [3]) and formal languages (e.g. [10]). As announced above, we represent an information base as a tree the structure of which is controlled by a formal grammar. We borrow the terms "scheme" and "instance" from the relational model and use the former to indicate the grammar and the latter to indicate the tree.

Definition 2.1

An information base scheme is a formal grammar $\mathcal{G} = (V, T, S, P)$ *with* V *a finite set of attributes,* T *a finite set of constants,* S *a set of axioms,* $S \subseteq V$*, and* P *a finite set of productions of the form* $A \rightarrow s$ *where* $A \in V$*,* $s \in (V \cup T)^+$*, and each attribute appears at most once in* s*.* ∎

Actual data will be represented in a tree the internal nodes of which are labeled by attributes. Note that we do not require the leaves to be labeled by constants; if a leaf is labeled by an attribute, this simply means there are no data known for that attribute.

Definition 2.2

Let $\mathcal{G} = (V, T, S, P)$ *be an information base scheme. A data-tree (D-tree) over* $\mathcal{G}$ *is a non-empty tree the nodes of which are labeled with elements of* $V \cup T$ *in such a way that each internal node is labeled by an attribute.* ∎

If $\mathbf{D}$ is a D-tree over some scheme $\mathcal{G}$, then $\mathbf{rt(D)}$ will denote its root. If $\mathbf{n}$ is a node of $\mathbf{D}$, then $\mathbf{prt(n)}$ will denote its parent (if existent), $\mathbf{chd(n)}$ the sequence of all its children, and $\mathbf{chtr(n)}$ the sequence of the subtrees of $\mathbf{D}$ the roots of which are the children of $\mathbf{n}$. Finally, $\mathrm{lbl}(\mathbf{n})$ will denote the label of $\mathbf{n}$ and $\mathrm{chd}(\mathbf{n})$ the sequence of the labels of all children of $\mathbf{n}$.

A D-tree over an information base scheme which is also a derivation tree over that grammar, will be called an information base instance.

Definition 2.3

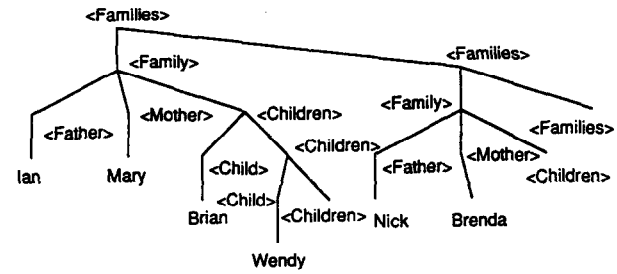
Let $\mathcal{G} = (V, T, S, P)$ *be an information base scheme. A D-tree* $\mathbf{D}$ *over* $\mathcal{G}$ *is called an information base instance over* $\mathcal{G}$ *if* $\mathrm{lbl}(\mathbf{rt(D)}) \in S$ *and for each internal node* $\mathbf{n}$ *there exists a production* $\mathrm{lbl}(\mathbf{n}) \rightarrow \mathrm{chd}(\mathbf{n})$ *in* P*.* ∎

Example 2.1

Consider the information base scheme $\mathcal{G} = (V, T, S, P)$ with:

- $V = \{Families, Family, Father, Mother, Children, Child, String, Char\}$;
- $T = \{A, \ldots, Z, a, \ldots, z\}$;
- $S = \{Families\}$;
- $P = \{\langle Families\rangle \rightarrow \langle Family\rangle\langle Families\rangle$
  $\langle Family\rangle \rightarrow \langle Father\rangle\langle Mother\rangle\langle Children\rangle$
  $\langle Children\rangle \rightarrow \langle Child\rangle\langle Children\rangle$
  $\langle Father\rangle \rightarrow \langle String\rangle$
  $\langle Mother\rangle \rightarrow \langle String\rangle$
  $\langle Child\rangle \rightarrow \langle String\rangle$
  $\langle String\rangle \rightarrow \langle Char\rangle\langle String\rangle$
  $\langle Char\rangle \rightarrow A, \ldots, \langle Char\rangle \rightarrow z$ $\}$

representing the structure on information base concerning families with their children. Note that this information base scheme also includes an "implementation" of strings. In the sequel, however, we will not bother with this low-level representation, and omit the corresponding productions, since this is not our main concern. With this in mind, the following D-tree represents an information base instance over $\mathcal{G}$:



∎

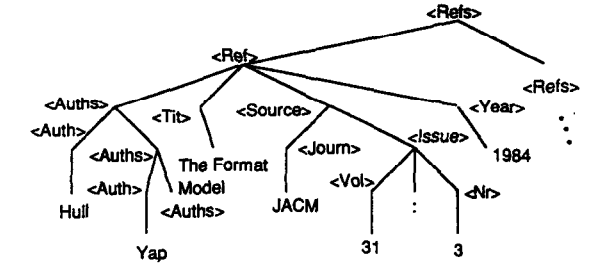
Grammar-based models turn out to be highly appropriate for representing text-dominated databases, as was observed by Gonnet and Tompa. The following example is inspired by [11].

Example 2.2

Consider an information base scheme with the following productions:

$\langle Refs\rangle \rightarrow \langle Ref\rangle\langle Refs\rangle$
$\langle Ref\rangle \rightarrow \langle Auths\rangle\langle Tit\rangle\langle Source\rangle\langle Year\rangle$
$\langle Auths\rangle \rightarrow \langle Auth\rangle\langle Auths\rangle$
$\langle Source\rangle \rightarrow \langle Journ\rangle\langle Issue\rangle$
$\langle Source\rangle \rightarrow \langle Book\rangle$
$\langle Issue\rangle \rightarrow \langle Vol\rangle : \langle Nr\rangle$

Part of an information base instance is shown below:



∎

In the grammatical model, the emphasis is obviously on the structure of a D-tree and its contents, rather than the actual nodes of which it consists. Therefore we define two D-trees $\mathbf{D}_1$ and $\mathbf{D}_2$ over some scheme $\mathcal{G}$ to be *isomorphic*, denoted $\mathbf{D}_1 \cong \mathbf{D}_2$, if there exists a mapping between the nodes of $\mathbf{D}_1$ and $\mathbf{D}_2$ which is one to one and onto, preserving both labels and the tree-structure. ∎

3. An algebra on information bases

In this section we propose an algebraic language for the manipulation of grammatically defined information bases, not only allowing to formulate queries but also to apply more general transformations. Each operator is defined both on scheme and on instance level. In the sequel, we implicitly assume that only one information base instance $\mathbf{D}$ over some scheme $\mathcal{G} = (V, T, S, P)$ is considered at a time.

First, we define three types of substitutions, which do not alter the structure of an information base instance, but only change (attribute) labels.

The parent substitution $\Sigma\pi[A \rightarrow s, B]$ substitutes by B all attributes A from which s is derived.

Definition 3.1

Let $A \rightarrow s \in P$. Let B be an attribute (B does not have to be in V) and suppose that A and B do never occur simultaneously in the right-hand side of a production of P. The parent substitution is defined as follows:

- $\Sigma\pi[A \rightarrow s, B](\mathcal{G}) = \mathcal{G}' = (V', T, S', P')$ *where:*
  - $V' = V \cup \{B\}$;
  - *if* $A \in S$, *then* $S' = S \cup \{B\}$, *else* $S' = S$;
  - *Let* $P'' = (P - \{A \rightarrow s\}) \cup \{B \rightarrow s\}$. *Then:* $P' = P'' \cup \{C \rightarrow s_1 B s_2 \mid C \rightarrow s_1 A s_2 \in P''\}$.
- $\Sigma\pi[A \rightarrow s, B](\mathbf{D})$ *is obtained by relabeling each node* $\mathbf{n}$ *in* $\mathbf{D}$ *with* $\mathrm{lbl}(\mathbf{n}) = A$ *and* $\mathrm{chd}(\mathbf{n}) = s$ *by* B. ∎

Note that the condition on A and B prevents B from appearing more than once in a production of $\mathcal{G}'$.

Now let $B \in V$ and let $A \rightarrow s_1 B s_2 \in P$. Let C be an attribute (C does not have to be in V) and suppose that C does not occur in $s_1 s_2$. The *child substitution* $\Sigma\chi[A \rightarrow s_1 B s_2, B, C]$ substitutes by C all attributes B in a string $s_1 B s_2$ which is derived from A. Because of the analogy with parent substitution, we omit a formal definition.

Finally, we define equality substitution. The equality substitution $\Sigma\epsilon[A \rightarrow s_1 B s_2, B \rightarrow s_3, C, D]$ substitutes by D all attributes B in a string $s_1 B s_2$ which is derived from A and from which s_3 is derived, provided B has both a sibling and a child labeled C that define isomorphic subtrees.

Definition 3.2

Let $A \rightarrow s_1 B s_2, B \rightarrow s_3 \in P$. Let $C \in V$ and suppose that C occurs both in $s_1 s_2$ and s_3. Let D be an attribute (D does not have to be in V) and suppose that D does not occur in $s_1 s_2$. The equality substitution is defined as follows:

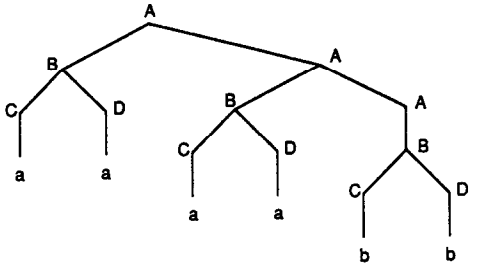
- $\Sigma\epsilon[A \rightarrow s_1 B s_2, B \rightarrow s_3, C, D](\mathcal{G}) = \mathcal{G}'$ *where:* $\mathcal{G}' = (V', T, S, P')$ *with:*
  - $V' = V \cup \{D\}$;
  - *if* $A = B$, *then:* $P' = P \cup \{A \rightarrow s_1 D s_2, D \rightarrow s_3, D \rightarrow s_1 D s_2\}$, *else* $P' = P \cup \{A \rightarrow s_1 D s_2, D \rightarrow s_3\}$.
- *Let* $\mathbf{n}$ *be a internal node in* $\mathbf{D}$ *with* $\mathrm{lbl}(\mathbf{n}) = B$, $\mathrm{lbl}(\mathbf{prt}(\mathbf{n})) = A$, $\mathrm{chd}(\mathbf{prt}(\mathbf{n})) = s_1 B s_2$, $\mathrm{chd}(\mathbf{n}) = s_3$. *Let* $\mathbf{m}_1$ *and* $\mathbf{m}_2$ *be the sibling respectively the child of* $\mathbf{n}$ *with label* C *and let* $\mathbf{D}_1$ *and* $\mathbf{D}_2$ *be the subtrees of* $\mathbf{D}$ *with* $\mathbf{rt}(\mathbf{D}_1) = \mathbf{m}_1$ *and* $\mathbf{rt}(\mathbf{D}_2) = \mathbf{m}_2$. $\Sigma\epsilon[A \rightarrow s_1 B s_2, B \rightarrow s_3, C, D](\mathbf{D})$ *is obtained by relabeling by* D *each such node* $\mathbf{n}$ *for which* $\mathbf{D}_1 \cong \mathbf{D}_2$. ∎

Example 3.1

Consider an information base scheme $\mathcal{G} = (V, T, S, P)$ with $V = \{A, B, C, D\}$, $T = \{a, b\}$, $S = \{A\}$ and

$$P = \{A \rightarrow BA, A \rightarrow B, B \rightarrow CD, C \rightarrow a, C \rightarrow b, D \rightarrow a, D \rightarrow b\}$$

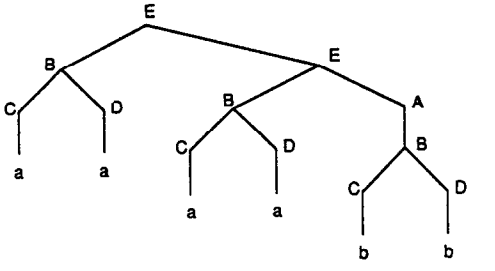
and let $\mathbf{D}$ be the following information base instance:



Then the parent substitution $\Sigma\pi[A \rightarrow BA, E]$ yields the information base scheme $\mathcal{G}' = (V', T, S', P')$ with $V' = \{A, B, C, D, E\}$, $S' = \{A, E\}$ and

$$P' = \{E \rightarrow BA, A \rightarrow B, B \rightarrow CD, C \rightarrow a, C \rightarrow b, D \rightarrow a, D \rightarrow b, E \rightarrow BE\}$$

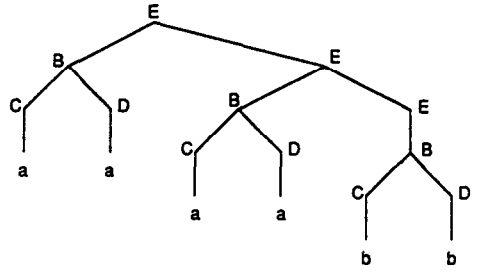
and the information base instance $\mathbf{D}'$:



The child substitution $\Sigma\chi[E \to BA, A, E]$ applied to the information base thus obtained, yields the information base scheme $\mathcal{G}'' = (V', T, S', P'')$ with

$$P'' = \{A \to B, B \to CD, C \to a, \\ C \to b, D \to a, D \to b, E \to BE, E \to B\}$$

and the information base instance $\mathbf{D}''$:



Finally, the equality substitution $\Sigma\epsilon[E \to BE, E \to BE, B, D]$ applied to the last result yields the information base scheme $\mathcal{G}''' = (V', T, S', P''')$ with

$$P''' = \{A \to B, B \to CD, C \to a, \\ C \to b, D \to a, D \to b, E \to BE, \\ E \to B, E \to BD, D \to BE, D \to BD\}$$

and the information base instance $\mathbf{D}''' = \mathbf{D}''$, since no node of $\mathbf{D}''$ labeled E satisfies all conditions required for relabeling by the given equality substitution. ∎

Next, we define two operators that allow the introduction of new nodes and the removal of existing ones.

The node insertion $N\iota[A \to s_1s_2s_3, s_1Bs_3]$ inserts in each derivation of $s_1s_2s_3$ from A a node B as a child of A and the father of s_2.

Definition 3.3

Let $A \to s_1s_2s_3 \in P$ with s_2 not empty. Let B be an attribute not in V. The node insertion is defined as follows:

- $N\iota[A \to s_1s_2s_3, s_1Bs_3](\mathcal{G}) = \mathcal{G}' = (V', T, S, P')$ *where:*
  - $V' = V \cup \{B\}$;
  - $P' = P - \{A \to s_1s_2s_3\} \cup \{A \to s_1Bs_3, B \to s_2\}$.
- *Let* $\mathbf{n}$ *be a node of* $\mathbf{D}$ *with* $\mathrm{lbl}(\mathbf{n}) = A$ *and* $\mathrm{chd}(\mathbf{n}) = s_1s_2s_3$. $N\iota[A \to s_1s_2s_3, s_1Bs_3](\mathbf{D})$ *is obtained by inserting for each such node* $\mathbf{n}$ *a node* $\mathbf{n}'$ *for which* $\mathrm{lbl}(\mathbf{n}') = B$, $\mathbf{prt}(\mathbf{n}') = \mathbf{n}$ *and* $\mathbf{chd}(\mathbf{n}')$ *is the subinterval of* $\mathbf{chd}(\mathbf{n})$ *corresponding to* s_2. ∎

The node deletion $N\delta[A]$ deletes each subtree the root of which is labeled by A.

Definition 3.4

Let $A \in V$. The node deletion is defined as follows:

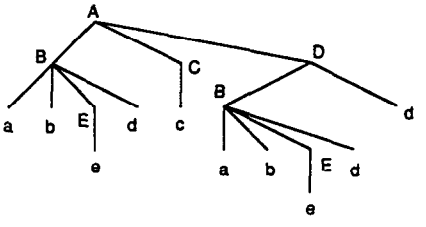
- $N\delta[A](\mathcal{G}) = \mathcal{G}' = (V', T, S', P'')$ *where:*
  - $V' = V - \{A\}$;
  - $S' = S - \{A\}$;
  - *Let* $P' = \{B \to s \mid B \to s \in P, B \neq A$ and A does not occur in $s\}$. *Then* $P'' = P' \cup \{B \to s_1s_2 \mid B \to s_1As_2 \in P$ and $B \neq A\}$.
- $N\delta[A](\mathbf{D})$ *is obtained by deleting each subtree* $\mathbf{D}'$ *from* $\mathbf{D}$ *with* $\mathrm{lbl}(\mathbf{rt}(\mathbf{D}')) = A$. ∎

Example 3.2

Consider an information base scheme $\mathcal{G} = (V, T, S, P)$ with $V = \{A, B, C, D, E\}$, $T = \{a, b, c, d, e\}$, $S = \{A\}$ and

$$P = \{A \to BCD, B \to abEd, C \to c, D \to Bd, E \to e\}$$

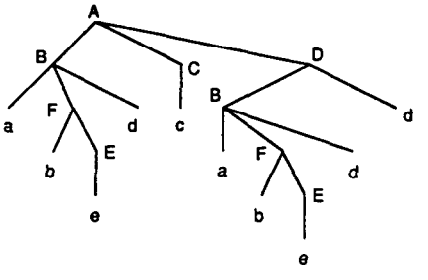
and let $\mathbf{D}$ be the following information base instance:



Then the node insertion $N\iota[B \to abEd, aFd]$ yields the information base scheme $\mathcal{G}' = (V', T, S, P')$ with $V' = \{A, B, C, D, E, F\}$ and

$$P' = \{A \to BCD, B \to aFd, C \to c, \\ D \to Bd, E \to e, F \to bE\}$$

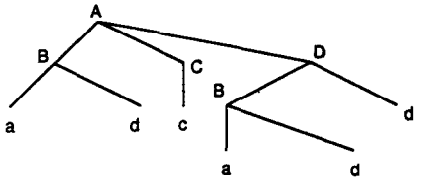
and the information base instance $\mathbf{D}'$:



The node deletion $N\delta[F]$ applied to the information base thus obtained, yields the information base scheme $\mathcal{G}'' = (V'', T, S, P'')$ with $V'' = \{A, B, C, D, E\}$ and

$$P'' = \{A \to BCD, B \to ad, C \to c, D \to Bd, E \to e\}$$

and the information base instance $\mathbf{D}''$:



Note that $N\delta[A]$ applied to any of the above information bases would yield the empty information base. ∎

We also need two operators that copy information from one place in an information base to another.

Definition 3.5

Let $\mathcal{G} = (V, T, S, P)$ be an information base scheme and let $\mathbf{D}$ *be an information base instance over $\mathcal{G}$. Let $A \rightarrow s_1 B s_2, B \rightarrow s_3 \in P$. Let $C \in V$ and suppose that C occurs in $s_1 s_2$. Let D be an attribute (D does not have to be in V) and suppose that D does not occur in s_3. The downward duplication is defined as follows:*

- $\Delta\delta[A \rightarrow s_1 B s_2, B \rightarrow s_3, C, D](\mathcal{G}) = \mathcal{G}'$ *where:*
  $\mathcal{G} = (V', T, S, P')$ *with:*
  - $V' = V \cup \{D\}$;
  - $P' = P \cup \{B \rightarrow s_3 D\} \cup \{D \rightarrow s \mid C \rightarrow s \in P\}$.
- *Let* $\mathbf{n}$ *be a internal node in* $\mathbf{D}$ *with* $\mathrm{lbl}(\mathbf{n}) = B$, $\mathrm{lbl}(\mathbf{prt}(\mathbf{n})) = A$, $\mathrm{chd}(\mathbf{prt}(\mathbf{n})) = s_1 B s_2$, $\mathrm{chd}(\mathbf{n}) = s_3$. *Suppose there is a node* $\mathbf{m}$ *in* $\mathbf{chd}(\mathbf{prt}(\mathbf{n}))$ *with* $\mathrm{lbl}(\mathbf{m}) = C$. $\Delta\delta[A \rightarrow s_1 B s_2, B \rightarrow s_3, C, D](\mathbf{D})$ *is obtained by adding to* $\mathbf{chd}(\mathbf{n})$ *for each such node* $\mathbf{n}$ *a rightmost sibling* $\mathbf{n}'$ *with* $\mathrm{lbl}(\mathbf{n}') = D$. *The subtree* $\mathbf{D}'$ *of* $\Delta\delta[A \rightarrow s_1 B s_2, B \rightarrow s_3, C, D](\mathbf{D})$ *defined by* $\mathbf{rt}(\mathbf{D}') = \mathbf{n}'$ *is isomorphic to* $\Delta\delta[A \rightarrow s_1 B s_2, B \rightarrow s_3, C, D](\mathbf{D}'')$, *where the D-tree* $\mathbf{D}''$ *is defined by* $\mathrm{lbl}(\mathbf{rt}(\mathbf{D}'')) = D$ *and* $\mathbf{chtr}(\mathbf{rt}(\mathbf{D}'')) = \mathbf{chtr}(\mathbf{m})$. ∎

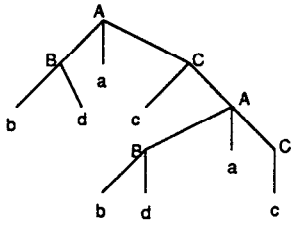
Suppose, for the sake of argument that C is in e.g. s_2 For applying this recursive definition sequentially, one has to proceed right-to-left. First, one searches for the leftmost nodes labeled A from which $s_1 B s_2$ is derived, such that s_3 is derived from B. Then add a rightmost sibling D to s_3. The subtree with root D is obtained by relabeling by D the root of an isomorphic copy of the subtree with root C in s_1. If this is done, the same procedure is repeated for the next node labeled A satisfying the same conditions as above. Repeating this until no further action is possible yields the desired result.

Example 3.3

Consider an information base scheme $\mathcal{G} = (V, T, S, P)$ with $V = \{A, B, C\}$, $T = \{a, b, c, d\}$, $S = \{A\}$ and

$$P = \{A \rightarrow BaC, B \rightarrow bd, C \rightarrow cA, C \rightarrow c\}$$

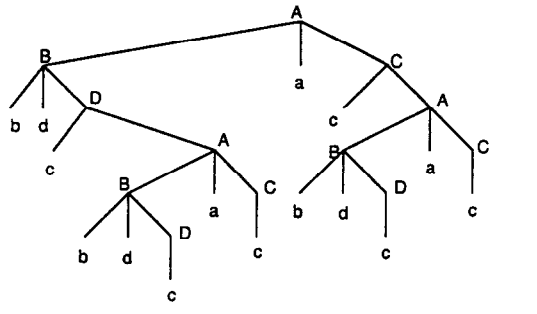
and let $\mathbf{D}$ be the following information base instance over $\mathcal{G}$:



Then the downward duplication $\Delta\delta[A \rightarrow BaC, B \rightarrow bd, C, D]$ yields the information base scheme $\mathcal{G}' = (V', T, S, P')$ with $V' = \{A, B, C, D\}$ and

$$P' = \{A \rightarrow BaC, B \rightarrow bdD, C \rightarrow cA, C \rightarrow c, D \rightarrow cA, D \rightarrow c\}$$

and the information base instance $\mathbf{D}'$:



∎

The downward duplication copies information downwards into the tree; its upward counterpart is called *upward duplication*. Let $A \rightarrow s_1 B s_2, B \rightarrow s_3 \in P$. Let $C \in V$ and suppose that C occurs in s_3. Let D be an attribute (D does not have to be in V) and suppose that D does not occur in $s_1 s_2$. The upward duplication $\Delta v[A \rightarrow s_1 B s_2, B \rightarrow s_3, C, D]$ adds a rightmost sibling D to $s_1 B s_2$ and transfers information under C in $s_1 s_2$ to D. Since upward duplication is analogous to downward duplication, we omit both the formal definition and an example.

Finally, we introduce a permutation. Basically, a permutation rearranges the children derived by some production $A \rightarrow s$.

Definition 3.6

Let $A \rightarrow s_1 \in P$ and let $s_2 \in (V \cup T)^+$ contain the same attributes as s_1. The permutation is defined as follows:

- $\Pi[A \rightarrow s_1, s_2](\mathcal{G}) = \mathcal{G}' = (V, T, S, P')$ *where:*

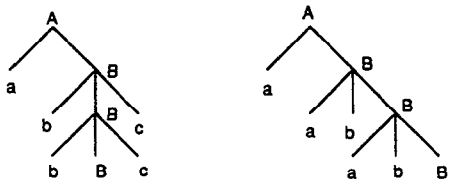
$$P' = (P - \{A \rightarrow s_1\}) \cup \{A \rightarrow s_2\}$$

- *Let* $\mathbf{n}$ *be a node in* $\mathbf{D}$ *with* $\mathrm{lbl}(\mathbf{n}) = A$ *and* $\mathrm{chd}(\mathbf{n}) = s_1$. $\Pi[A \rightarrow s_1, s_2](\mathbf{D})$ *is obtained by substituting new nodes for* $\mathbf{chd}(\mathbf{n})$ *such that* $\mathrm{chd}(\mathbf{n})$ *becomes* s_2. *Let B be an attribute in s_2 and let* $\mathbf{m}$ *be the node in* $\Pi[A \rightarrow s_1, s_2](\mathbf{D})$ *with* $\mathrm{lbl}(\mathbf{m}) = B$ *and* $\mathbf{prt}(\mathbf{m}) = \mathbf{n}$. *Then the subtree* $\mathbf{D}'$ *of* $\Pi[A \rightarrow s_1, s_2](\mathbf{D})$ *defined by* $\mathbf{rt}(\mathbf{D}') = \mathbf{m}$ *is isomorphic to* $\Pi[A \rightarrow s_1, s_2](\mathbf{D}'')$, *where the D-tree* $\mathbf{D}''$ *is the subtree of* $\mathbf{D}$ *defined by* $\mathrm{lbl}(\mathbf{rt}(\mathbf{D})) = B$ *and* $\mathbf{prt}(\mathbf{rt}(\mathbf{D})) = \mathbf{n}$. ∎

Our notion of permutation is somewhat wider than what is usually understood by this term. A permutation does indeed permute attributes, but can also insert, delete, and rearrange constants.

Example 3.4

Consider an information base scheme $\mathcal{G} = (V, T, S, P)$ with $V = \{A, B\}$, $T = \{a, b, c\}$, $S = \{A\}$ and $P = \{A \rightarrow aB, B \rightarrow bBc\}$ and let $\mathbf{D}$ be the information base instance over $\mathcal{G}$ represented by the tree below to the left:



Then the permutation $\Pi[B \rightarrow bBc, abB]$ yields the information base scheme $\mathcal{G}' = (V, T, S, P')$ with $P = \{A \rightarrow aB, B \rightarrow abB\}$ and the information base instance $\mathbf{D}'$ represented by the tree above to the right. ∎

The eight operators presented above define the *grammatical algebra*. Many other conceivable operators can be expressed in terms of these eight. By "expressed" we mean there is a sequence of instance-independent algebra operations that returns the same result *at the instance level*. In general, it is unavoidable that the scheme returned by the algebra sequence defines a larger language than the scheme returned by the original operator.

As an example, we consider the *node merging*, an operator often needed in practical applications. The node merging $N\mu[A \rightarrow s_1Bs_2, B \rightarrow s_3]$ is obtained by pruning out each attribute B in a string s_1Bs_2 which is derived from A and from which s_3 is derived. In this way, $s_1s_3s_2$ will be derived from A instead. Clearly, a node insertion $N\iota[A \rightarrow s_1s_2s_3, s_1Bs_3]$ can be undone by the node merging $N\mu[A \rightarrow s_1Bs_3, B \rightarrow s_2]$. We now show:

Theorem 3.1

Node merging can be expressed in the grammatical algebra.

Proof: Rather than giving a notationally cumbersome proof, we illustrate the general techniques that are needed on an example. Let $V = \{A, B, C\}$, $T = \{a, b, c\}$, $S = \{A\}$ and $P = \{A \rightarrow aBc, B \rightarrow bC, C \rightarrow c\}$. $N\mu[A \rightarrow aBc, B \rightarrow bC]$ can be expressed by consecutively performing the following operations:

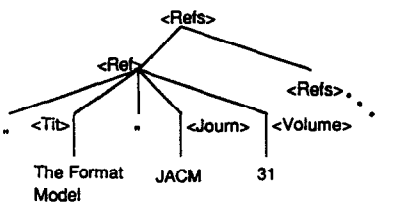
1. the upward duplication
$$\Delta\upsilon[A \rightarrow aBc, B \rightarrow bC, C, C];$$
2. the equality substitution
$$\Sigma\epsilon[A \rightarrow aBcC, B \rightarrow bC, C, B'];$$
3. the node deletion $N\delta[B']$;
4. the permutation $\Pi[A \rightarrow acC, abCc]$.

We invite the reader to check our claim on a concrete instance. ∎

We now return to the bibliographical Example 2.2 to illustrate how the grammatical algebra can be used to solve queries or to perform transformations on a more realistic information base.

Example 3.5

Reconsider the information base of Example 2.2. Suppose we want to extract only the information on journal titles (between double quotes) with the name of the journal and the volume. The instance of Example 2.2 would then be transformed into:



The transformation can be accomplished by consecutively performing the following operations:

1. $N\delta[\langle Auths\rangle]$;
2. $N\delta[\langle Year\rangle]$;
3. $\Delta\upsilon[\langle Source\rangle \rightarrow \langle Journ\rangle\langle Issue\rangle,$
$\langle Issue\rangle \rightarrow \langle Vol\rangle : \langle Nr\rangle, \langle Vol\rangle, \langle Volume\rangle]$;
4. $N\delta[\langle Issue\rangle]$;
5. $N\mu[\langle Ref\rangle \rightarrow \langle Tit\rangle\langle Source\rangle,$
$\langle Source\rangle \rightarrow \langle Journ\rangle\langle Volume\rangle]$;
6. $\Pi[\langle Ref\rangle \rightarrow \langle Tit\rangle\langle Journ\rangle\langle Volume\rangle,$
$''\langle Tit\rangle''\langle Journ\rangle\langle Volume\rangle]$. ∎

Observe that the instance obtained in the above example can be considered as a representation of a flat relational database model relation in the grammatical model. Below, we show how unary relational algebra operators can be performed. Note that, in order to simulate union, difference and join, we need binary operators on information bases. This, however, is beyond the scope of the present paper.

Example 3.6

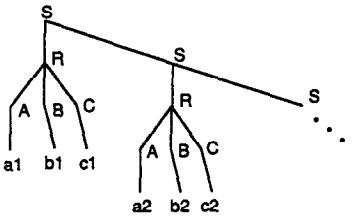
Consider the following relational database relation R:

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
	⋮	

This relation can be represented as an information base with scheme $\mathcal{G} = (V, T, S, P)$ where $V = \{S, R, A, B, C\}$, $T = \{a_1, \ldots, b_1, \ldots, c_1, \ldots\}$, $S = \{R\}$ and:

$$P = \{S \rightarrow RS, R \rightarrow ABC, A \rightarrow a_1, \ldots, B \rightarrow b_1, \ldots, C \rightarrow c_1, \ldots\}$$

and the information base instance $\mathbf{D}$:



- The *renaming* of A to A' can be expressed as the child substitution $\Sigma\chi[R \rightarrow ABC, A, A']$.
- The *projection* of R onto AB can be expressed as the node deletion $N\delta[C]$.

- The *selection* $A = C$ can be expressed by consecutively performing:
  1. the node insertion $\mathrm{N}\iota[R \rightarrow ABC, ABC']$;
  2. the child substitution $\Sigma\chi[C' \rightarrow C, C, A]$;
  3. the equality substitution $\Sigma\epsilon[R \rightarrow ABC', C' \rightarrow A, A, C'']$;
  4. the parent substitution $\Sigma\pi[R \rightarrow ABC', R']$;
  5. the node deletion $\mathrm{N}\delta[R']$;
  6. the child substitution $\Sigma\chi[C'' \rightarrow A, A, C]$.
  7. the node merging $\mathrm{N}\mu[R \rightarrow ABC''R, C'' \rightarrow C]$;

If desired, redundant S-nodes can be removed by repeatedly applying the node mergings $\mathrm{N}\mu[S \rightarrow S, S \rightarrow RS]$ and $\mathrm{N}\mu[S \rightarrow RS, S \rightarrow S]$. ∎

4. A calculus on information bases

In this section, we present a more logic-oriented transformation language, inspired by the relational calculus. An expression in the grammatical calculus we propose, consists of a set of conditions and a transformation clause, both build from some variables. Informally, applying a calculus expression to an information base means performing the required transformations for each "occurrence" of the variables satisfying the set of conditions. Since the variables in a calculus expression represent so-called *rootless data trees*, we first explain this notion.

Definition 4.1

Let $\mathcal{G} = (V, T, S, P)$ be an information base scheme. A rootless data tree (R-tree) is a finite sequence of D-trees such that each attribute is the label of the root of at most one D-tree in the sequence. The set of all R-trees over $\mathcal{G}$ is denoted $\mathcal{R}(\mathcal{G})$. ∎

Note that the notion of isomorphic D-trees can be extended in a natural way to R-trees. We have to introduce the following notations concerning R-trees:

Definition 4.2

Let $\mathcal{G} = (V, T, S, P)$ be an information base scheme and let $\mathbf{R}$ be an R-tree over $\mathcal{G}$, then:

- $\mathbf{R} = (\mathbf{D}_1, \ldots, \mathbf{D}_n)$ *denotes the sequence of D-trees of which* $\mathbf{R}$ *consists;*
- $\mathbf{top}(\mathbf{R}) = (\mathbf{rt}(\mathbf{D}_1), \ldots, \mathbf{rt}(\mathbf{D}_n))$ *denotes the sequence of the roots of the D-trees of which* $\mathbf{R}$ *consists, and* $\mathrm{top}(\mathbf{R}) = \mathrm{lbl}(\mathbf{rt}(\mathbf{D}_1)) \ldots \mathrm{lbl}(\mathbf{rt}(\mathbf{D}_n))$ *denotes the corresponding sequence of labels;*
- *the empty R-tree is denoted* $\mathbf{E}$. ∎

R-trees can be contained in D-trees:

Definition 4.3

Let $\mathcal{G} = (V, T, S, P)$ be an information base scheme. Let $\mathbf{D}$ be a D-tree and $\mathbf{R}$ be an R-tree over $\mathcal{G}$. $\mathbf{R}$ is called a rootless subtree of $\mathbf{D}$ if $\mathbf{R} = (\mathbf{D}_1, \ldots, \mathbf{D}_n)$, $\mathbf{D}_1, \ldots, \mathbf{D}_n$ are subtrees of $\mathbf{D}$ and $\mathbf{top}(\mathbf{R})$ is a sequence of consecutive siblings of $\mathbf{D}$. The set of all rootless subtrees of $\mathbf{D}$ is denoted $\mathbf{rst}(\mathbf{D})$. ∎

Finally, we need two simple operations on R-trees:

Definition 4.4

Let $\mathcal{G} = (V, T, S, P)$ be an information base scheme.

- *Let $\mathbf{R}_1 = (\mathbf{D}_1, \ldots, \mathbf{D}_m)$ and $\mathbf{R}_2 = (\mathbf{D}_{m+1}, \ldots, \mathbf{D}_n)$ be R-trees over $\mathcal{G}$ such that $\mathrm{top}(\mathbf{R}_1)$ and $\mathrm{top}(\mathbf{R}_2)$ have no attributes in common. Then the concatenation $\mathbf{R}_1\mathbf{R}_2$ is the R-tree defined by $\mathbf{R}_1\mathbf{R}_2 = (\mathbf{D}_1, \ldots, \mathbf{D}_n)$.*
- *Let $\mathbf{R}$ be an R-tree over $\mathcal{G}$ and let $\mathbf{n}$ be an arbitrary node. If either $\mathbf{R}$ is empty or $\mathrm{lbl}(\mathbf{n})$ is an attribute, then the completion $\mathbf{nR}$ is a D-tree, defined by $\mathbf{rt}(\mathbf{nR}) = \mathbf{n}$ and $\mathbf{chtr}(\mathbf{nR}) = \mathbf{R}$.* ∎

As mentioned, variables in a calculus expression represent rootless subtrees of the information base instance under consideration. From these variables, terms are build using concatenation and completion, representing in turn R-trees. The set of conditions in a calculus expression is a set of declarations of variables by terms, possibly augmented with some equations between variables. The substitution clause also consists of a variable and a term, by which the variable has to substituted. Of course, the variables in the substitution clause have to occur in the set of conditions. Before formalizing the syntax of a grammatical calculus expression, we clarify the concept by an example.

Example 4.1

Reconsider the information base of Example 2.2 and the query of Example 3.5, described in the grammatical algebra. This query can also be solved by the following grammatical calculus expression:

$$
\begin{aligned}
[\rho_2 \leftarrow'' \; & (\langle Tit\rangle\rho_4)''(\langle Journ\rangle\rho_7)(\langle Volume\rangle\rho_9) \mid \\
& \{\rho_1 := (\langle Ref\rangle\rho_2) \\
& \;\rho_2 := (\langle Auths\rangle\rho_3)(\langle Tit\rangle\rho_4) \\
& \;(\langle Source\rangle\rho_5)(\langle Year\rangle\rho_6) \\
& \;\rho_5 := (\langle Journ\rangle\rho_7)(\langle Issue\rangle\rho_8) \\
& \;\rho_8 := (\langle Vol\rangle\rho_9) : (\langle Nr\rangle\rho_{10}) \qquad \}]
\end{aligned}
$$

∎

We now formally define the syntax of the grammatical calculus. Throughout this exposition, we assume that $\mathcal{V} = \{\rho_i \mid i = 1, 2, 3, \ldots\}$ is an infinitely enumerable set of *variables*.

Definition 4.5

Let $\mathcal{G} = (V, T, S, P)$ be an information base scheme.

- *A basic term over $\mathcal{G}$ has one of the following three types:*
  - *type 1: a $(a \in T)$;*
  - *type 2: ρ_i $(\rho_i \in \mathcal{V})$;*
  - *type 3: $(A\rho_i)$ $(A \in V,\ \rho_i \in \mathcal{V})$.*
- *A term over $\mathcal{G}$ is a finite sequence of basic terms over $\mathcal{G}$ that contains at most one basic term of type 2 and in which each variable and each attribute appears at most once. The empty term is denoted ε. The set of all variables occurring in a term t is denoted* $\mathrm{var}(t)$. ∎

Definition 4.6

Let $\mathcal{G} = (V, T, S, P)$ be an information base scheme.

- *A declaration over $\mathcal{G}$ has the form $\rho_i := t$ with $\rho_i \in \mathcal{V}$ and t a term over $\mathcal{G}$ in which ρ_i does not occur.*
- *An equation over $\mathcal{G}$ has the form $\rho_i = \rho_j$ with $\rho_i, \rho_j \in \mathcal{V}$.* ∎

Definition 4.7

Let $\mathcal{G} = (V, T, S, P)$ be an information base scheme. Let $\mathcal{D} = \{\rho_i := t_i \mid i \in I\}$, I a set of indices, be a finite set of declarations over $\mathcal{G}$ in which no variable appears in the right-hand side of more than one equation. Let var($\mathcal{D}$) *denote the set of all variables occurring in $\mathcal{D}$. Consider the associated directed graph $\mathcal{G}(\mathcal{D})$ with set of nodes* var($\mathcal{D}$) *and set of edges $\{\rho_j \rightarrow \rho_k \mid j \in I \wedge \rho_k \in \text{var}(t_j)\}$. $\mathcal{D}$ is called hierarchical if $\mathcal{G}(\mathcal{D})$ is a tree, and, furthermore, the root ρ_{root} has a declaration of the form $\rho_{root} := (A\rho_l)$ for some $A \in V$ and $\rho_l \in \mathcal{V}$.* ∎

Note that, by definition, a hierarchical set of declarations has at most one declaration for each variable.

In order to ensure the definition of an expression to be semantically meaningful, we need to define an order on the variables in a hierarchical set of declarations:

Definition 4.8

Let $\mathcal{G} = (V, T, S, P)$ be an information base scheme and let $\mathcal{D}$ be a hierarchical set of declarations over $\mathcal{G}$. Let $\mathcal{G}(\mathcal{D})$ be the associated tree. Let ρ_i and ρ_j be two nodes in this tree. ρ_j is said to be to the left (right) of ρ_j if ρ_i precedes (follows) ρ_j in the post-order (pre-order). ∎

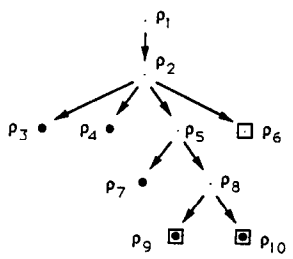
Clearly, the descendants of a node in $\mathcal{G}(\mathcal{D})$ can be characterized as the only nodes that are both to the left and to the right of that node.

Example 4.2

Reconsider the set of declarations in the right-hand side of the expression in Example 4.1:

$$\begin{aligned}\{&\rho_1 := (\langle Ref\rangle\rho_2)\\ &\rho_2 := (\langle Auths\rangle\rho_3)(\langle Tit\rangle\rho_4)(\langle Source\rangle\rho_5)(\langle Year\rangle\rho_6)\\ &\rho_5 := (\langle Journ\rangle\rho_7)(\langle Issue\rangle\rho_8)\\ &\rho_8 := (\langle Vol\rangle\rho_9) : (\langle Nr\rangle\rho_{10}) \qquad \}\end{aligned}$$

Obviously, this is a hierarchical set of declarations with associated tree:



The nodes to the left of ρ_8 are marked with "•"; the nodes to the right of ρ_8 are marked with "□". ∎

We can now define an expression:

Definition 4.9

Let $\mathcal{G} = (V, T, S, P)$ be an information base scheme. An expression over $\mathcal{G}$ has the form $[\rho_j \leftarrow u \mid \mathcal{D} \cup \mathcal{E}]$ with $\rho_j \in \mathcal{V}$, u a term over $\mathcal{G}$, $\mathcal{D}$ a hierarchical set of declarations over $\mathcal{G}$ and $\mathcal{E}$ a set of equations over $\mathcal{G}$, satisfying:

- *all variables in the expression occur in $\mathcal{D}$;*
- *the variables in u are either all to the left or all to the right of ρ_j in $\mathcal{G}(\mathcal{D})$;*
- *if, in addition, ρ_j is the root of $\mathcal{G}(\mathcal{D})$, then $u = (A\rho_m)$ for some $A \in V$ and $\rho_m \in \mathcal{V}$ or $u = \varepsilon$.* ∎

We invite the reader to check that the expression (without equations) in Example 4.1 satisfies Definition 4.9.

Before formally defining the semantics of a grammatical calculus expression, we show how the grammatical algebra operators can be expressed in the calculus.

Example 4.3

Reconsider the Examples 3.1 to 3.4.

- The parent substitution $\Sigma\pi[A \rightarrow BA, E]$ can be expressed by
  $[\rho_1 \leftarrow (E\rho_2) \mid \{\rho_1 := (A\rho_2), \rho_2 := (B\rho_3)(A\rho_4)\}]$.
- The child substitution $\Sigma\epsilon[E \rightarrow BA, A, E]$ can be expressed by $[\rho_2 \leftarrow (B\rho_3)(E\rho_4) \mid \{\rho_1 := (E\rho_2), \rho_2 := (B\rho_3)(A\rho_4)\}]$.
- The equality substitution
  $\Sigma\epsilon[E \rightarrow BE, E \rightarrow BE, B, D]$ can be expressed by
  $[\rho_2 \leftarrow (B\rho_3)(D\rho_4) \mid \{\rho_1 := (E\rho_2), \rho_2 := (B\rho_3)(E\rho_4), \rho_4 := (B\rho_5)(E\rho_6), \rho_3 = \rho_5\}]$.
- The node insertion $\text{N}\iota[B \rightarrow abEd, aFd]$ can be expressed by $[\rho_2 \leftarrow (F\rho_3)d \mid \{\rho_1 := (B\rho_2), \rho_2 := a\rho_3 d, \rho_3 := b(E\rho_4)\}]$.
- The node deletion $\text{N}\delta[F]$ can be expressed by $[\rho_1 \leftarrow \varepsilon \mid \rho_1 := (F\rho_2)\}]$.
- The downward duplication
  $\Delta\delta[A \rightarrow BaC, B \rightarrow bd, C, D]$
  can be expressed by $[\rho_3 := bd(D\rho_4) \mid \{\rho_1 := (A\rho_2), \rho_2 := (B\rho_3)a(C\rho_4), \rho_3 := bd\}]$.
- The permutation $\Pi[B \rightarrow bBc, abB]$ can be expressed by
  $[\rho_2 \leftarrow ab(B\rho_3) \mid \{\rho_1 := (B\rho_2), \rho_2 := b(B\rho_3)c\}]$. ∎

Describing the semantics of the grammatical calculus should consist of two parts: explaining what happens with schemes and explaining what happens with instances. Since, as observed earlier in this paper, it is unrealistic to compare information base operations at the scheme level, we shall not elaborate on how calculus expressions work on information base schemes. It is nevertheless possible to apply calculus expressions to the schemes as well [12].

We now formally define the semantics of a grammatical calculus expression at the instance level. Though conceptually simple, the formalism itself is rather involved. This stems mainly from the fact that when rearranging subtrees in applying a calculus expression, one must be able to describe how this rearrangement is "propagated" downwards into these subtrees.

Evaluating a calculus expression on an information base instance is done in two stages:

1. First, the variables in an expression are "valuated" as rootless subtrees of the considered information base instance, satisfying the declarations and equations in that expression;
2. Then, the D-tree representing the information base instance is transformed according to these valuation and the transformation rule in the left-hand side of the expression.

The first stage is described in Definition 4.10; the second one in Definition 4.11.

Definition 4.10

Let $\mathcal{G} = (V, T, S, P)$ be an information base scheme and let $\mathbf{D}$ be a information base instance over $\mathcal{G}$. Let $\mathcal{D}$ be a hierarchical set of declarations and $\mathcal{E}$ a set of equations over $\mathcal{G}$. Let f: $\mathrm{var}(\mathcal{D}) \rightarrow \mathbf{rst}(\mathbf{D})$ be a total mapping from variables in $\mathcal{D}$ to rootless subtrees of $\mathbf{D}$. f is called a valuation of $\mathcal{D}$ and $\mathcal{E}$ in $\mathbf{D}$ if:

- *for each declaration $\rho_i := \varepsilon$ in $\mathcal{D}$, $f(\rho_i) = \mathbf{E}$;*
- *for each declaration $\rho_i := t = t_1 \ldots t_k$ in $\mathcal{D}$, $t_1, \ldots, t_k$ being basic terms, $f(\rho_i) = \mathbf{R}_1 \ldots \mathbf{R}_k$ with for each $j = 1, \ldots, k$:*
  - *if $t_j = a$ for some $a \in T$, then $\mathbf{R}_j$ is a one node rootless subtree labeled a,*
  - *if $t_j = \rho_k$ for some $\rho_k \in \mathcal{V}$, then $\mathbf{R}_j = f(\rho_k)$,*
  - *if $t_j = (A\rho_k)$ for some $A \in V$ and $\rho_k \in \mathcal{V}$, then there is a node $\mathbf{n}$ in $\mathbf{D}$ with $\mathrm{lbl}(\mathbf{n}) = A$ such that $\mathbf{R}_j = (\mathbf{n} f(\rho_k))$,*
- *for each equation $\rho_i = \rho_j$ in $\mathcal{E}$, $f(\rho_i) \cong f(\rho_j)$.*

The set of all valuations of $\mathcal{D}$ and $\mathcal{E}$ in $\mathbf{D}$ is denoted $\mathcal{F}(\mathcal{D} \cup \mathcal{E}, \mathbf{D})$. ∎

As said before, it is our intention to define the result of a calculus expression by performing a transformation on the information base instance under consideration for each valuation of its hierarchical set of declarations and set of equations. This strategy works, because of:

Lemma 4.1

Let $\mathcal{G} = (V, T, S, P)$ be an information base scheme, let $\mathcal{D}$ be a hierarchical set of equations over $\mathcal{G}$, let $\mathcal{E}$ be a set of equations over $\mathcal{G}$ and let $\mathbf{D}$ be a information base instance over $\mathcal{G}$. Let ρ_i be an arbitrary variable in $\mathcal{D}$ and let $\mathbf{R}$ be a rootless subtree of $\mathbf{D}$. There exists at most one valuation f of $\mathcal{D}$ and $\mathcal{E}$ in $\mathbf{D}$ such that $f(\rho_i)$ is an interval of $\mathbf{R}$. ∎

We now define a transformation of an information base by an expression:

Definition 4.11

Let $\mathcal{G} = (V, T, S, P)$ be an information base scheme and let $\mathbf{D}$ be an information base instance over $\mathcal{G}$. Let $E \equiv [\rho_j \leftarrow u \mid \mathcal{D} \cup \mathcal{E}]$ be an expression over $\mathcal{G}$. A total mapping g: $\mathbf{rst}(\mathbf{D}) \rightarrow \mathcal{R}(\mathcal{G})/_{\cong}$ from the rootless subtrees of $\mathbf{D}$ to classes of isomorphic rootless trees is a transformation of $\mathbf{D}$ by E if it satisfies the following conditions:[1]

- *$g(\mathbf{E}) = \mathbf{E}$;*
- *Let $\mathbf{R} \in \mathbf{rst}(\mathbf{D})$ be a one-node rootless tree such that for no valuation $f \in \mathcal{F}(\mathcal{D} \cup \mathcal{E}, \mathbf{D})$, $f(\rho_j) = \mathbf{R}$. Then $g(\mathbf{R}) \cong \mathbf{R}$;*
- *Let $(\mathbf{nR}) \in \mathbf{rst}(\mathbf{D})$ be a rootless subtree such that for no valuation $f \in \mathcal{F}(\mathcal{D} \cup \mathcal{E}, \mathbf{D})$, $f(\rho_j) = (\mathbf{nR})$. Then $g(\mathbf{nR}) \cong (\mathbf{n}g(\mathbf{R}))$;*
- *Let $\mathbf{R} \in \mathbf{rst}(\mathbf{D})$ be a rootless subtree such that for no valuation $f \in \mathcal{F}(\mathcal{D} \cup \mathcal{E}, \mathbf{D})$, $f(\rho_j)$ is an interval of $\mathbf{R}$. Then if $\mathbf{R} = \mathbf{R}_1\mathbf{R}_2$, $g(\mathbf{R}) \cong g(\mathbf{R}_1)g(\mathbf{R}_2)$;*
- *Let $\mathbf{R} \in \mathbf{rst}(\mathbf{D})$ be a rootless subtree such that for some valuation $f \in \mathcal{F}(\mathcal{D} \cup \mathcal{E}, \mathbf{D})$, $f(\rho_j) = \mathbf{R}$. Let $u = u_1 \ldots u_n$ with $u_1, \ldots, u_n$ basic terms. Then $g(\mathbf{R}) \cong \mathbf{R}_1 \ldots \mathbf{R}_n$ with, for $k = 1, \ldots, n$:*
  - *If $u_k = a$ for some $a \in T$, then $\mathbf{R}_k$ is a one-node rootless tree labeled a,*
  - *if $u_k = \rho_l$ for some $\rho_l \in \mathcal{V}$, then $\mathbf{R}_k \cong g(f(\rho_l))$,*
  - *if $u_k = (A\rho_l)$ for some $A \in V$ and $\rho_l \in \mathcal{V}$, then $\mathbf{R}_k \cong \mathbf{nR}'$ with $\mathrm{lbl}(\mathbf{n}) = A$ and $\mathbf{R}' \cong g(f(\rho_l))$;*
- *Let $\mathbf{R} = \mathbf{R}_1\mathbf{R}_2\mathbf{R}_3 \in \mathbf{rst}(\mathbf{D})$ be a rootless subtree such that for some valuation $f \in \mathcal{F}(\mathcal{D} \cup \mathcal{E}, \mathbf{D})$, $f(\rho_j) = \mathbf{R}_2$. Then $g(\mathbf{R}) \cong g(\mathbf{R}_1)g(\mathbf{R}_2)g(\mathbf{R}_3)$.* ∎

A transformation of an information base instance defines a new D-tree as is shown by:

Theorem 4.1

Let $\mathcal{G} = (V, T, S, P)$ be an information base scheme and let $\mathbf{D}$ be an information base instance over $\mathcal{G}$. Let E be an expression over $\mathcal{G}$ and let g be a transformation of $\mathbf{D}$ by E. Then there exists a D-tree $\mathbf{D}' \in \mathcal{D}(\mathcal{G})$ such that $g((\mathbf{D})) \cong (\mathbf{D}')$. ∎

Now suppose an expression $E \equiv [\rho_j \leftarrow u \mid \mathcal{D} \cup \mathcal{E}]$ over some scheme $\mathcal{G}$ is given. If $\mathbf{D}$ is an information base over $\mathcal{G}$, then the conditions of Definition 4.9 guarantee there exists a unique transformation g of $\mathbf{D}$ by E that can be constructed "deterministically", depending on the case, either left-to-right or right-to-left. Obviously, this transformation defines a new D-tree $\mathbf{D}'$ satisfying $g((\mathbf{D})) \cong \mathbf{D}'$. The *result $E(\mathbf{D})$ of the calculus expression E applied to $\mathbf{D}$* is now defined as (the class of) $\mathbf{D}'$, if $\mathbf{D}'$ is in turn an information base instance over $\mathcal{G}$, and undefined otherwise.

We now presented two languages for transforming information bases. Inspired by the classical result in the relational model, we were able to show:

Theorem 4.2 [12]

The grammatical algebra and grammatical calculus are equivalent with regard to expressive power. ∎

The equivalence between grammatical algebra and calculus obviously gives some "naturalness" to both languages, although some language independent criterion should still be looked for.

[1] For reasons of convenience, the conditions are formulated as if $g(\mathbf{R})$ were an arbitrary representation of the class under consideration.

5. Conclusions and future work

In this paper a simple model for representing the hierarchical structure in information is proposed. Two equivalent methods for querying in this data model are given. The expressive power of these languages, however, is not yet clear. In particular, it is not known how they are related to querying facilities in other data models, that can be simulated by the grammatical model. Furthermore we are looking for a well-adapted interface that is integrated in a more general environment. It is remarkable that there seems to be no fundamental distinction between updates and querying in this model. Other aspects such as transforming several given trees into one result tree, constraint checking and implementation strategies are under investigation.

Though it can be considered as an extension of the relational model, the grammatical model, being hierarchical in nature, is of course not suited for all database applications. In particular, the notion of "shared component" is difficult to express in a tree. It is therefore interesting to look as well for a characterization of the semantic expressiveness of the grammatical model. On the other hand, one could also look for "network-like" extensions of this model, using the theory of graph-grammars (e.g. [16]).

References

[1] Abiteboul S., N. Bidoit, "Non First Normal Form Relations to Represent Hierarchically Organized Data", *Proc. 3th PODS*, Waterloo, Ont., 1984, pp. 191–198.

[2] Abiteboul S., R. Hull, "IFO: A Formal Semantic Database Model", *Trans. Database Systems* 12:4, December 1987, pp. 525–565.

[3] Aho A., J. Hopcroft, J.D. Ullman, "Data Structures and Algorithms", Addison-Wesley, 1983.

[4] Beeri C. et al., "Sets and Negation in a Logic Database Language (LDL1)", *Proc. 6th PODS*, San Diego, 1987, pp. 21–37.

[5] Bidoit N., "The Verso Algebra or How to Answer Queries with Fewer Joins", *Journ. Computer and System Sciences* 35:3, 1987, pp. 321–364.

[6] Chen P., "The Entity-Relationship Model: Toward a Unified View of Data", *Trans. Database Systems 1:1*, 1976, pp. 9–36.

[7] Codd E.F., "Extending the Database Relational Model to Capture More Meaning", *Trans. Database Systems* 4:4, 1979, pp. 397–434.

[8] Colby L.S. *pers. comm.*, Nov. 1988.

[9] Dadam P. et al.,"A DBMS Prototype to Support Extended NF2 Relations: An Integrated View on Flat Tables and Hierarchies", *SIGMOD Conf.*, San Francisco, 1986, pp. 356–364.

[10] Ginsburg S., "The Mathematical Theory of Context-Free Languages", McGraw-Hill, 1966.

[11] Gonnet G.H., F.W. Tompa, "Mind Your Grammar: a New Approach to Modelling Text", *Techn. Rep.*, Univ. Waterloo, 1988.

[12] Gyssens M., J. Paredaens, D. Van Gucht, "A Grammar-Based Approach towards Unifying Hierarchical Data Models", *Techn. Rep.* 88-33, Univ. Antwerp, 1988.

[13] Hammer M.,D. McLeod, Database description with SDM: A Semantic Database Model", *Trans. Database Systems* 6:3, 1981, pp. 351–386.

[14] Hull R., C. Yap, "The Format Model: A Theory of Database Organization", *Journ. ACM* 31:3, 1984, pp. 518–537.

[15] Jaeschke G., H.-J. Schek, "Remarks on the Algebra on Non First Normal Form Relations", *2nd PODS*, Atlanta, 1982, pp. 124–138.

[16] Janssens D., G. Rozenberg, "On the Structure of Node Label Controlled Languages", *Information Sciences* 20, 1980, pp. 191–216.

[17] Kersten M.L., A.P.J.M. Siebes, C.A. van den Berg, "Using a Graph Rewriting System for Databases", *Techn. Rep.* CS-R8805, Centre for Math. and Comp. Science, Amsterdam, 1988.

[18] Kuper G.M., M.Y. Vardi, "A New Approach to Database Logic", *3rd PODS*, Waterloo, Ont., 1984, pp. 124–138.

[19] Maier D., "The Theory of Relational Databases", Computer Science Press, 1983.

[20] Paredaens J.,P. De Bra, M. Gyssens, D. Van Gucht, "The Structure of the Relational Database Model", *EATCS Monographs on Theoretical Computer Science*, Springer Verlag, 1989.

[21] Ridjanovic D., M.L. Brodie, "Defining Database Dynamics with Attribute Grammars", *Inf. Processing Letters* 14:3, 1982, pp. 132–138.

[22] Shipman D., "The Functional Data Model and the Data Language DAPLEX", *Trans. Database Systems* 6:10, 1981, pp. 140–173.

[23] Thomas S.J., P.C. Fischer, "Nested Relational Structures", *Advances in Computing Research III: The Theory of Databases*, P. Kanellakis (ed.), JAI Press, 1986, pp. 269–307.

[24] Ullman J.D., "Principles of Database and Knowledge-Base Systems, Vol I", Computer Science Pr., 1988.