MTA SZTAKI Közlemények 37/1987 pp. 185-200

SEARCHING AND RETRIEVAL IN DATABASES BY TREES

H. THIELE

Department of Mathematics
Humboldt-University at Berlin
POB 1297, Berlin, 1086, GDR

1. Introduction

The present paper deals with a logical approach to searching and retrieval in (relational) databases by trees. We start with a many-sorted first-order language with functional symbols, predicate symbols and the symbol = for the identity. Then, using the notion of signature, we define databases or, in algebraic terms, heterogeneous (operational-relational) algebraic systems. For trees defined on the basis of the many-sorted first-order language we define two kinds of semantics: trees working in identification mode and trees working in retrieval mode. For both of these semantics we introduce the notions soundness, completeness and (partial) equivalence of trees and relating to these notions we formulate theorems of decidability and axiomatizability. The paper concludes with two theorems giving a basis for a (fast) retrieval algorithm using a proof system.

In the following we use the usual set theoretic notions and notations, in particular, if $\mathfrak{S}$ is a collection of sets, then $\bigcup \mathfrak{S}$ denotes the union of all sets $S \in \mathfrak{S}$. The empty set is denoted by ϕ. If S is an arbitrary set, then S^* is the set of all finite sequences of elements of S. As symbol for the empty sequence we use e. A binary relation R over S is an equivalence relation if it is reflexive over S, symmetric and transitive. R is said to be a partial equivalence relation if it is symmetric and transitive only.

2. Signatures and syntax of many-sorted first-order languages

The basis of many-sorted first-order languages and of interpreting structures is the notion of signature.

DEFINITION 2.1.

A quadruple $\Sigma = [S,F,P,TYPE]$ is said to be a signature if and only if S, F, P are pairwise disjoint sets and TYPE is a mapping from $F \cup P$ into S^*, where $TYPE(f) \neq e$ if $f \in F$.

REMARKS.

1. The set S, F, P is called the set of sorts, the set of functional symbols, and the set of predicate symbols, respectively, of the signature Σ. Sorts, functional symbols, and predicate symbols are denoted by s, f, p, respectively, possibly with arbitrary indices.

2. If for $s_1,\ldots,s_n,s \in S$ $(n \geqq 0)$, $f \in F$ and $p \in P$ the equation $TYPE(f) = s_1\ldots s_n s$ and $TYPE(p) = s_1\ldots s_n$, respectively, holds, then $s_1\ldots s_n s$ and $s_1\ldots s_n$ is called the type of f and p, respectively. In the case of the functional symbol f we introduce the input type and the output type of f by the definition $ITYPE(f) =_{def} s_1\ldots s_n$ and $OTYPE(f) =_{def} s$. The output type s of f is also called output sort of f.

3. If for $f \in F$ the input type of f is empty then we call f an individual name (of the sort s if $OTYPE(f) = s$). Σ is said to be operational and relational, respectively, if and only if P is empty and F contains only individual names, respectively.

DEFINITION 2.2.

Given a fixed signature $\Sigma = [S,F,P,TYPE]$. To every sort $s \in S$ we introduce a fixed infinite set V_s of individual variables of the sort s denoted by v^s, x^s, y^s, z^s (possibly with additional indices). We define $V =_{def} \bigcup_{s \in S} V_s$. Starting with the variables of V, with the functional and predicate symbols of Σ and using the logical connections $\sim$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\forall$, $\exists$, the sign = for the identity, the comma , and the brackets ( ) we form terms t (possibly with indices) and formulae A, B, C (possibly with indices) as usual for many-sorted first-order languages.

Because of lack of space we do not explicitely formulate this definition. The many-sorted first-order language defined above is denoted by $\mathcal{L}(\Sigma,V)$.

3. Databases and semantics of many-sorted first-order languages

Firstly, we define the notion of Σ-database (or, in algebraic terms, the notion of Σ-algebraic system).

DEFINITION 3.1.

A quadruple $\mathcal{D} = [\Sigma, \Delta, \Phi, \Pi]$ is said to be a Σ-database (or shortly a database) if and only if the following conditions are satisfied:

1. Σ is a signature, say $\Sigma = [S,F,P,TYPE]$.
2. Δ is a mapping assigning a set Δ_s to every sort $s \in S$.
3. Φ is a mapping assigning a partial function Φ_f from $\Delta_{s_1} \times \ldots \times \Delta_{s_n}$ into Δ_s to every functional symbol $f \in F$ where $TYPE(f) = s_1 \ldots s_n s$ and $n \geq 0$.
4. Π is a mapping assigning a total function Π_p from $\Delta_{s_1} \times \ldots \times \Delta_{s_n}$ into $\{0,1\}$ to every predicate symbol $p \in P$ where $TYPE(p) = s_1 \ldots s_n$ and $n \geq 0$.

REMARKS.

1. The set Δ_s is called the domain of the sort $s \in S$ (with respect to the database $\mathcal{D}$). It can be empty. The elements of Δ_s are called individuals of the sort s and denoted by ξ^s.
2. If P is empty, then $\mathcal{D}$ is called operational. If, additionally, for every functional symbol $f \in F$, the function Φ_f is total, then $\mathcal{D}$ is called total-operational.
3. If F contains only individual names (i.e. nullary functional symbols), then $\mathcal{D}$ is called a relational database.
4. An individual name $\nu \in F$ is said to be $\mathcal{D}$-proper if and only if the (nullary) function Φ_ν is total, i.e. Φ_ν is an element of Δ_s if $OTYPE(\nu) = s$.
5. The case of relations with nullvalues can be built in as follows. For every sort $s \in S$ we adjoin a new element $null^s$ to Δ_s where $null^s \notin \Delta_s$. Then we replace the term

$\Delta_{s_1} \times \ldots \times \Delta_{s_n}$ by $(\Delta_{s_1} \cup \{\text{null}^{s_1}\}) \times \ldots \times (\Delta_{s_n} \cup \{\text{null}^{s_n}\})$ in condition 4 of DEFINITION 3.1. A predicate $\overline{\Pi}_p$ is said to be null-free if and only if for every $\xi^{s_1} \in \Delta_s \cup \{\text{null}^{s_1}\}, \ldots,$ $\xi^{s_n} \in \Delta_{s_n} \cup \{\text{null}^{s_n}\}$, the equation $\overline{\Pi}_p(\xi^{s_1}, \ldots \xi^{s_n}) = 1$ implies $\xi^{s_1} \in \Delta_{s_1}, \ldots, \xi^{s_n} \in \Delta_{s_n}$.

6. The notion of database introduced by DEFINITION 3.1 does not formalize any mechanism of accessibility. But that is even not necessary for the following investigations.

Now, given an arbitrary database $\mathcal{D} = [\Sigma, \Delta, \Phi, \overline{\Pi}]$, where $\Sigma = [S,F,P,\text{TYPE}]$, and a $\mathcal{D}$-evaluation σ, i. e. for every sort $s \in S$, σ is a mapping from the set V_s of all individual variables of the sort s into the set Δ_s of all individuals of the sort s. Let t and A be an arbitrary term and an arbitrary formula of the language $\mathcal{L}(\Sigma,V)$. Then by structural induction on t and A we get the following.

DEFINITION 3.2.

1. $\text{EL}(t,\mathcal{D},\sigma) =_{\text{def}}$ the element of the sort $\text{OTYPE}(t)$ which is assigned to the term t with respect to the interpreting database $\mathcal{D}$ and the $\mathcal{D}$-evaluation σ.

2. $\text{VAL}(A,\mathcal{D},\sigma) =_{\text{def}}$ the logical value of $\{0,1\}$ which is assigned to formula A with respect to the interpreting database $\mathcal{D}$ and the $\mathcal{D}$-evaluation σ.

Because of lack of space we cannot fully formulate these definitions, but we will accentuate the following facts.

REMARKS.

1. If the domain Δ_s is empty, then the evaluation σ is not defined for any individual variable of the sort s.

2. As the mapping σ or the function Φ_f, where $f \in F$, can be partial, the functional EL can be partial, too.

3. For equations $t_1 = t_2$, where t_1 and t_2 are terms of the same sort, we define $\text{VAL}(t_1=t_2,\mathcal{D},\sigma) =_{\text{def}} 1$ if and only if $\text{EL}(t_1,\mathcal{D},\sigma)$ and $\text{EL}(t_2,\mathcal{D},\sigma)$ exist and $\text{EL}(t_1,\mathcal{D},\sigma)=\text{EL}(t_2,\mathcal{D},\sigma)$. Note that this interpretation agrees with the interpretation of so-called "existence-equations" $t_1 \overset{e}{=} t_2$ (see [1,15]). Predicate

formulae $pt_1 \ldots t_n$ are analogously interpreted.

4. If only the individual variable x^s freely occurs in the term t and in the formula A then we say that t and A, respectively, is one-dimensional of the sort s. In this case the element $EL(t, \mathfrak{D}, \sigma)$ and the logical value $VAL(A, \mathfrak{D}, \sigma)$ depends only on $\sigma(x^s)$. Therefore we can denote this element and this value by $EL(t, \mathfrak{D}, \xi^s)$ and $VAL(A, \mathfrak{D}, \xi^s)$, respectively, if $\sigma(x^s) = \xi^s$.

For a given Σ-database $\mathfrak{D}$ and a collection Γ of Σ-database we define as usual:

DEFINITION 3.3.

1. A is a $\mathfrak{D}$-theorem (shortly $\mathfrak{D} \models A$) if and only if for every $\mathfrak{D}$-evaluation σ, $VAL(A, \mathfrak{D}\sigma) = 1$.

2. A is a Γ-theorem (shortly $\Gamma \models A$) if and only if for every database $\mathfrak{D} \in \Gamma$, A is a $\mathfrak{D}$-theorem.

4. The syntax of trees

Let $\Sigma = [S, F, P, TYPE]$ be an arbitrary signature. Consider the many-sorted first-order language $\mathcal{L}(\Sigma, V)$. Now, we introduce the "new" symbol ε, i.e. a symbol not occurring among the symbols of $\mathcal{L}(\Sigma, V)$. It is called the empty name and in contrast to individual names it cannot carry any semantic meaning (with respect to a given Σ-database $\mathfrak{D}$). Extended individual names of the sort s are defined as individual names of the sort s or the empty name ε. For an arbitrary term t the output sort of t is defined as the sort of t if t is an individual variable and as the sort $OTYPE(f)$ if t begins with f.

By the following four generation rules we define trees of the sort $s \in S$. They are denoted by T (possibly with arbitrary indices).

DEFINITION 4.1.

1. If ν is an extended individual name of the sort s, then ν is a tree of the sort s.

2. If A is a one-dimensional formula of the sort s and T_0, T_1 are trees of the sort s, then $A(T_0, T_1)$ is a tree of

the sort s.

3. If T_1, T_2 are trees of the sort s, then (T_1,T_2) is a tree of the sort s.

4. If t is a one-dimensional term of the sort s and the output sort s' and if for $m \geq 1$ the symbols $\nu_1,\ldots,\nu_m$ denote individual names of the sort s' and $T_1,\ldots,T_m$ are trees of the sort s, then $t(\nu_1:T_1,\ldots,\nu_m:T_m)$ is a tree of the sort s.

5. Trees working in identification mode

Now, we define the first interpretation of trees which is called "trees working in identification mode". Given an arbitrary database $\mathfrak{D} = [\Sigma, \Delta, \Phi, \Pi]$, where $\Sigma = [S,F,P,TYPE]$, and a collection Γ of Σ-databases. Furthermore let T be a tree of the sort $s \in S$ and $\xi^s \in \Delta_s$. By structural induction on T we define the set $SEIN(T, \mathfrak{D}, \xi^s)$ of extended individual names of the sort s which is computed by the tree T with respect to the interpreting database $\mathfrak{D}$ and the input element ξ^s.

DEFINITION 5.1.

1. $SEIN(T, \mathfrak{D}, \xi^s) =_{def} \{T\}$ if T is an extended individual name (of the sort s)
2. $SEIN(A(T_o,T_1), \mathfrak{D}, \xi^s)$

$$=_{def} \begin{cases} SEIN(T_o, \mathfrak{D}, \xi^s) & \text{if } VAL(A, \mathfrak{D}, \xi^s) = 0 \\ SEIN(T_1, \mathfrak{D}, \xi^s) & \text{if } VAL(A, \mathfrak{D}, \xi^s) = 1 \end{cases}$$

3. $SEIN((T_1,T_2), \mathfrak{D}, \xi^s) =_{def} SEIN(T_1, \mathfrak{D}, \xi^s) \cup SEIN(T_2, \mathfrak{D}, \xi^2)$
4. $SEIN(t(\nu_1:T_1,\ldots,\nu_m:T_m), \mathfrak{D}, \xi^s)$

$$=_{def} \bigcup \{SEIN(T_\mu, \mathfrak{D}, \xi^s) \mid \mu \in \{1,\ldots,m\},\ \Phi_{\nu_\mu} \text{ is total, } EL(t, \mathfrak{D}, \xi^s) \text{ exists, and } \Phi_{\nu_\mu} = EL(t, \mathfrak{D}, \xi^2)$$

REMARKS.

1. Using the functional SEIN we can interpret a tree working in identification mode as an identification algorithm as follows. Given an "unknown" element $\xi^s \in \Delta_s$, i.e. the name of ξ^s (or any name if several exist) is unknown. To know this element means to know some (or all) $\mathfrak{D}$-proper individual names of ξ^s, i.e. functional symbols $f \in F$ with $TYPE(f) = s$ such

that $\bar{\Phi}_f$ is total and $\bar{\Phi}_f = \xi^s$.

2. In applications we want to use only such trees T that do not compute "wrong" individual names ν, i.e. $\mathcal{D}$-proper individual names with $\bar{\Phi}_\nu \neq \xi^s$ or ν is the symbol ε. This idea leads to two notions of soundness for trees working in identification mode.

3. The claim to know all ($\mathcal{D}$-proper individual) names of an element $\xi^s \in \Delta_s$ leads to two notions of completeness for trees T working in identification mode.

4. Trees of the sort s generated by the rules 1 and 2 of DEFINITION 4.1 are called binary (or logical or deterministic) trees. As the interpretation shows binary trees working in identification mode compute at least one extended individual name, i.e. they cannot catch the phenomenon of synonymy.

5. DEFINITION 5.1.3 shows that the tree construct (T_1, T_2) can be interpreted as an unconditioned branching of the searching algorithm described by (T_1, T_2). This construct gives the possibility for syntactically describing multiple effects.

6. If for $\xi^s \in \Delta_s$ and $\mu \in \{1,\ldots,m\}$ there is no $\mathcal{D}$-proper individual name ν_μ such that $EL(t, \mathcal{D}, \xi^s)$ exists and $\bar{\Phi}_{\nu_\mu} = EL(t, \mathcal{D}, \xi^s)$, then the set $SEIN(t(\nu_1:T_1,\ldots,\nu_m:T_m), \mathcal{D}, \xi^s)$ is empty.

7. If for $\mu_1, \mu_2 \in \{1,\ldots,m\}$ with $\mu_1 \neq \mu_2$ the individual names ν_{μ_1} and ν_{μ_2} are equal, then we can interpret this fact as a syntactical branching of the searching algorithm described by the tree $t(\nu_1:T_1,\ldots,\nu_m:T_m)$. We will speak of semantic branching in this tree (with respect to $\mathcal{D}$ and ξ^s) if for $\mu \in \{1,\ldots,m\}$ there is more than one $\mathcal{D}$-proper individual name ν_μ such that $EL(t, \mathcal{D}, \xi^s)$ exists and $\bar{\Phi}_{\nu_\mu} = EL(t, \mathcal{D}, \xi^s)$.

DEFINITION 5.2.

1. T is said to be sound with respect to $\mathcal{D}$ if and only if for every $\xi^s \in \Delta_s$ and for every $\mathcal{D}$-proper individual name ν (of the sort s), if $\nu \in SEIN(T, \mathcal{D}, \xi^s)$ then $\bar{\Phi}_\nu = \xi^s$.

2. T is said to be strongly sound with respect to $\mathcal{D}$ if and only if T is sound with respect to $\mathcal{D}$ and for every $\xi^s \in \Delta_s$, $SEIN(T, \mathcal{D}, \xi^s)$ does not contain the empty name ε.

3. T is said to be complete with respect to $\mathcal{D}$ if and only if for every $\xi^s \in \Delta_s$ there exists a $\mathcal{D}$-proper individual name ν of the sort s such that $\Phi_\nu = \xi^s$ and $\nu \in SEIN(T, \mathcal{D}, \xi^s)$.

4. T is said to be strongly complete with respect to $\mathcal{D}$ if and only if T is complete with respect to T and for every $\xi^s \in \Delta_s$ and for every $\mathcal{D}$-proper individual name ν of the sort s, if $\Phi_\nu = \xi^s$ then $\nu \in SEIN(T, \mathcal{D}, \xi^s)$.

5. T is said to be sound, ..., strongly complete with respect to Γ if and only if for every database $\mathcal{D} \in \Gamma$, T is sound, ..., strongly complete with respect to $\mathcal{D}$, respectively.

6. If Γ consists of all arbitrary Σ-databases, then we say that T is logically sound, ..., logically strongly complete, respectively, and omit the symbol Γ.

We now define some (partial) equivalence relations for trees working in identification mode. These can be used, for instance, to develop theories of optimization or of inductive inference for trees (working in identification mode).

Given a database $\mathcal{D} = [\Sigma, \Delta, \Phi, \Pi]$ and a collection Γ of databases with the signature Σ. Let T_1 and T_2 be trees of the sort $s \in S$.

DEFINITION 5.3.

1. $T_1 \overset{\sim}{\mathcal{D}} T_2$ if and only if for every $\xi^s \in \Delta_s$ and for every $\mathcal{D}$-proper individual name ν of the sort s, $\nu \in SEIN(T_1, \mathcal{D}, \xi^s)$ if and only if $\nu \in SEIN(T_2, \mathcal{D}, \xi^s)$.

2. $T_1 \overset{\approx}{\mathcal{D}} T_2$ if and only if for every $\xi^s \in \Delta_s$,

$$SEIN(T_1, \mathcal{D}, \xi^s) = SEIN(T_2, \mathcal{D}, \xi^s)$$

3. $T_1 \overset{\approx}{\underset{\mathcal{D}}{\approx}} T_2$ if and only if $T_1 \overset{\approx}{\mathcal{D}} T_2$ and for every $\xi^s \in \Delta_s$, $SEIN(T_1, \mathcal{D}, \xi^s) \cup SEIN(T_2, \mathcal{D}, \xi^s)$ contains only $\mathcal{D}$-proper individual names.

4. $T_1 \overset{\sim}{\Gamma} T_2$, $T_1 \overset{\approx}{\Gamma} T_2$ and $T_1 \overset{\approx}{\underset{\Gamma}{\approx}} T_2$ if and only if for every database $\mathcal{D} \in \Gamma$, $T_1 \overset{\sim}{\mathcal{D}} T_2$, $T_1 \overset{\approx}{\mathcal{D}} T_2$ and $T_1 \overset{\approx}{\underset{\mathcal{D}}{\approx}} T_2$, respectively.

5. If Γ consists of all arbitrary Σ-databases, then we speak of logical equivalence of trees and omit the symbol Γ.

CONCLUSION.

1. $T_1 \overset{\approx}{\underset{\mathfrak{D}}{\approx}} T_2 \Rightarrow T_1 \overset{\approx}{\underset{\mathfrak{D}}{}} T_2 \Rightarrow T_1 \overset{\sim}{\underset{\mathfrak{D}}{}} T_2$

2. $T_1 \overset{\approx}{\underset{\Gamma}{\approx}} T_2 \Rightarrow T_1 \overset{\approx}{\underset{\Gamma}{}} T_2 \Rightarrow T_1 \overset{\sim}{\underset{\Gamma}{}} T_2$

3. $\overset{\sim}{\underset{\mathfrak{D}}{}}$, $\overset{\approx}{\underset{\mathfrak{D}}{}}$, $\overset{\sim}{\underset{\Gamma}{}}$, and $\overset{\approx}{\underset{\Gamma}{}}$ are reflexive, symmetric and transitive, i.e. equivalence relations.

4. $\overset{\approx}{\underset{\mathfrak{D}}{\approx}}$ and $\overset{\approx}{\underset{\Gamma}{\approx}}$ are symmetric and transitive, i.e. so-called partial equivalence relations.

PROOF.

Use definition 5.3.

THEOREM 5.1.

1. If the set of Γ-theorems of the language $\mathcal{L}(\Sigma,V)$ is decidable (recursively enumerable), then the sets of trees characterized by DEFINITION 5.2.5 and the binary relations for trees characterized by DEFINITION 5.3.4 are decidable (recursively enumerable) too, respectively.

2. If DS is a deductive system (possibly with ω-rules) for characterizing the Γ-theorems of $\mathcal{L}(\Sigma,V)$, then for each set of trees characterized by DEFINITION 5.2.5 and for each binary relation for trees characterized by DEFINITION 5.3.4, from DS in a natural way a deductive system (possibly with ω-rules, too) can be derived to characterize this set and this relation, respectively.

PROOF.

1. Use the definitions under consideration.

2. See the full draft version of this paper which will be published later elsewhere.

6. Trees working in retrieval mode

Now, we turn over to define retrieval processes described by trees.

Given a one-dimensional formula A and a tree T, both of the sort $s \in S$. We interpret A as a query or as a request expression of the given database $\mathfrak{D}$, i.e. by definition, we are interested in knowing at least one (or all) proper individual names of every element $\xi^s \in \Delta_s$ with $VAL(A,\mathfrak{D},\xi^s) = 1$.

Therefore we define the set $RET(T, \mathfrak{D}, A)$ of all individual names of the sort s retrieved in the tree T by the query A with respect to the database $\mathfrak{D}$ as follows.

DEFINITION 6.1.

$$RET(T, \mathfrak{D}, A) =_{def} \bigcup \{SEIN(T, \mathfrak{D}, \xi^s) \mid \xi^s \in \Delta_s \text{ and } VAL(A, \mathfrak{D}, \xi^s) = 1\}$$

Let $\mathfrak{R}$ be an arbitrary fixed set of one-dimensional formulae of the sort $s \in S$ from $\mathfrak{L}(\Sigma, V)$. In the following a set of this kind is called request language because only formulae of $\mathfrak{R}$ are admitted as queries. Let x^s be the individual variable (of the sort s) which freely occurs in A.

Analogously to DEFINITION 5.1 we formulate

DEFINITION 6.2.

1. T is said to be retrieval-sound with respect to $\mathfrak{D}$ and $\mathfrak{R}$ if and only if for every $A \in \mathfrak{R}$ and for every $\mathfrak{D}$-proper individual name ν of the sort s, if $\nu \in RET(T, \mathfrak{D}, A)$ then $\mathfrak{D} \models A(^x/_\nu)$.

2. T is said to be strongly retrieval-sound with respect to $\mathfrak{D}$ and $\mathfrak{R}$ if and only if T is retrieval sound and for every $A \in \mathfrak{R}$, the empty name ϵ is not an element of $RET(T, \mathfrak{D}, A)$.

3. T is said to be retrieval-complete with respect to $\mathfrak{D}$ and $\mathfrak{R}$ if and only if for every $A \in \mathfrak{R}$ and $\xi^s \in A_s$, if $VAL(A, \mathfrak{D}, \xi^s) = 1$, then there exists a $\mathfrak{D}$-proper individual name ν of the sort s such that $\Phi_\nu = \xi^s$ and $\nu \in RET(T, \mathfrak{D}, A)$.

4. T is said to be strongly retrieval-complete with respect to $\mathfrak{D}$ and $\mathfrak{R}$ if and only if T is retrieval complete with respect to $\mathfrak{D}$ and $\mathfrak{R}$ and for every $A \in \mathfrak{R}$, for every $\xi^s \in \Delta_s$ and for every $\mathfrak{D}$-proper individual name ν of the sort s, if $VAL(A, \mathfrak{D}, \xi^s) = 1$ and $\Phi_\nu = \xi^s$, then $\nu \in RET(T, \mathfrak{D}, A)$.

5. T is said to be retrieval-sound, ..., strongly retrieval-complete with respect to Γ and $\mathfrak{R}$ if and only if for every database $\mathfrak{D} \in \Gamma$, T is retrieval-sound, ..., strongly retrieval-complete with respect to $\mathfrak{D}$ and $\mathfrak{R}$, respectively.

6. If Γ consists of all arbitrary databases (with the signature Σ), then we say that T is logically retrieval-sound,

..., logically strongly retrieval-complete, respectively, and omit the symbol Γ.

Now, analogously to DEFINITION 5.3, we define the following binary relations for trees working in retrieval mode.

DEFINITION 6.3.

1. $T_1 \widetilde{\mathcal{D},\mathcal{R}} T_2$ if and only if for every $A \in \mathcal{R}$ and for every $\mathcal{D}$-proper individual name of the sort ν, $\nu \in \mathrm{RET}(T_1, \mathcal{D}, A)$ if and only if $\nu \in \mathrm{RET}(T_2, \mathcal{D}, A)$.

2. $T_1 \widetilde{\widetilde{\mathcal{D},\mathcal{R}}} T_2$ if and only if for every $A \in \mathcal{R}$,

$\mathrm{RET}(T_1, \mathcal{D}, A) = \mathrm{RET}(T_2, \mathcal{D}, A)$.

3. $T_1 \widetilde{\widetilde{\mathcal{D},\mathcal{R}}} T_2$ if and only if $T_1 \widetilde{\mathcal{D},\mathcal{R}} T_2$ and for every $A \in \mathcal{R}$, $\mathrm{RET}(T_1, \mathcal{D}, A) \cup \mathrm{RET}(T_2, \mathcal{D}, A)$ contains only $\mathcal{D}$-proper individual names.

4. $T_1 \widetilde{\Gamma,\mathcal{R}} T_2$, $T_1 \widetilde{\widetilde{\Gamma,\mathcal{R}}} T_2$ and $T_1 \widetilde{\widetilde{\Gamma,\mathcal{R}}} T_2$ if and only if for every database $\mathcal{D} \in \Gamma$, $T_1 \widetilde{\mathcal{D},\mathcal{R}} T_2$, $T_1 \widetilde{\mathcal{D},\mathcal{R}} T_2$ and $T_1 \widetilde{\widetilde{\mathcal{D},\mathcal{R}}} T_2$, respectively.

5. If Γ consists of all arbitrary databases with the signature Σ, then we speak of the logical equivalence of trees and omit the symbol Γ.

CONCLUSION

1. $T_1 \widetilde{\widetilde{\mathcal{D},\mathcal{R}}} T_2 \Longrightarrow T_1 \widetilde{\widetilde{\mathcal{D},\mathcal{R}}} T_2 \Longrightarrow T_1 \widetilde{\mathcal{D},\mathcal{R}} T_2$

2. $T_1 \widetilde{\widetilde{\Gamma,\mathcal{R}}} T_2 \Longrightarrow T_1 \widetilde{\widetilde{\Gamma,\mathcal{R}}} T_2 \Longrightarrow T_1 \widetilde{\Gamma,\mathcal{R}} T_2$

3. $\widetilde{\mathcal{D},\mathcal{R}}$, $\widetilde{\widetilde{\mathcal{D},\mathcal{R}}}$, $\widetilde{\Gamma,\mathcal{R}}$ and $\widetilde{\widetilde{\Gamma,\mathcal{R}}}$ are reflexive, symmetric and transitive, i.e. equivalence relations.

4. $\widetilde{\widetilde{\mathcal{D},\mathcal{R}}}$ and $\widetilde{\widetilde{\Gamma,\mathcal{R}}}$ are symmetric and transitive, i.e. so-called partial equivalence relations.

PROOF

Use definition 6.3.

Analogously to THEOREM 5.1 we can prove the following theorem.

THEOREM 6.1.

1. If the set of Γ-theorems of the language $\mathcal{L}(\Sigma, V)$ is decidable (recursively enumerable), then the sets of trees characterized by DEFINITION 6.2.5 and the binary relations for

trees characterized by definition 6.3.4 are decidable (recursively enumerable), too, respectively.

2. If DS is a deductive system (possibly with ω-rules) for characterizing the Γ-theorems of $\mathcal{L}_\Sigma$, then for each set of trees characterized by DEFINITION 6.2.5 and for each binary relation for trees characterized by DEFINITION 6.3.4 from DS in a natural way a deductive system (possibly with ω-rules, too) can be derived to characterize this set and this relation, respectively.

PROOF

1. Use the definitions under consideration.

2. See the full draft version of this paper which will be published later elsewhere.

7. A "logical" retrieval algorithm based on a proof system

By studying the functional properties of the mapping RET we can develop a "logical" retrieval algorithm which is based on a theorem tester or a proof system. The basis of this algorithm is the following theorem formulated with respect to the syntactical definition of trees. Assume that in A, B and t the same individual variable freely occurs.

THEOREM 7.1.

1. $\mathrm{RET}(\nu, \mathfrak{D}, A) = \left\{ \begin{array}{ll} \phi & \text{if } \mathfrak{D} \models \sim A \\ \{\nu\} & \text{if not } \mathfrak{D} \models \sim A \end{array} \right\}$ and ν is an extended individual name.

2. $\mathrm{RET}(B(T_0, T_1), \mathfrak{D}, A) = \mathrm{RET}(T_0, \mathfrak{D}, A \wedge \sim B) \cup \mathrm{RET}(T_1, \mathfrak{D}, A \wedge B)$

3. $\mathrm{RET}((T_1, T_2), \mathfrak{D}, A) = \mathrm{RET}(T_1, \mathfrak{D}, A) \cup \mathrm{RET}(T_2, \mathfrak{D}, A)$

4. $\mathrm{RET}(t(\nu_1 : T_1, \ldots, \nu_m : T_m), \mathfrak{D}, A) = \bigcup_{\mu=1}^{m} \mathrm{RET}(T_\mu, \mathfrak{D}, A \wedge t = \nu_\mu)$

PROOF

Use definition 6.1.

To describe the algorithm we have in mind we need the following two mappings WAY(T) and COND(T,ω), where $\omega \in \mathrm{WAY}(T)$. The first is to be the set of all ways of T, i.e. "connected" sequences of edges of T starting in the root of T and ending in a leaf of T. COND(T,ω) describes the logical condition derived in T along the way $\omega \in \mathrm{WAY}(T)$.

DEFINITION 7.1.

1. $WAY(\nu) =_{def} \{e\}$ if ν is an extended individual name.
2. $WAY(A(T_o,T_1)) =_{def} \{0\omega \mid \omega \in WAY(T_o)\} \cup \{1\omega \mid \omega \in WAY(T_1)\}$
3. $WAY((T_1,T_2)) =_{def} \{1\omega \mid \omega \in WAY(T_1)\} \cup \{2\omega \mid \omega \in WAY(T_2)\}$
4. $WAY(t(\nu_1:T_1,\ldots,\nu_m:T_m)) =_{def} \bigcup_{\mu=1}^{m} \{\mu\omega \mid \omega \in WAY(T_\mu)\}$

DEFINITION 7.2.

1. $COND(\nu,e) =_{def} A_o \rightarrow A_o$ if ν is an extended name, A_o is a fixed arbitrary formula.
2.0. $COND(A(T_o,T_1),0\omega) =_{def} \sim A \wedge COND(T_o,\omega)$
2.1. $COND(A(T_o,T_1),1\omega) =_{def} A \wedge COND(T_1,\omega)$
3.1. $COND((T_1,T_2),1\omega) =_{def} COND(T_1,\omega)$
3.2. $COND((T_1,T_2),2\omega) =_{def} COND(T_2,\omega)$
4. $COND(t(\nu_1:T_1,\ldots,\nu_m:T_m),\mu\omega) =_{def} t=\nu_\mu \wedge COND(T_\mu,\omega)$ if $\mu \in \{1,\ldots,m\}$.

Furthermore, it is clear how the extended individual name $EIN(T,\omega)$ determined as the label of the leaf which is given in the tree T by the way ω has to be defined. Using THEOREM 7.1 and the above definitions we can compute the set $RET(T,\mathfrak{D},A)$ by completely examining the set $WAY(T)$ and checking whether $\mathfrak{D} \models \sim(A \wedge COND(T,\omega))$ or not for $\omega \in WAY(T)$. By THEOREM 7.1 we know that for every extended individual name ν, $\nu \in RET(T,\mathfrak{D},A)$ if and only if there exists a way $\omega \in WAY(T)$ such that $\mathfrak{D} \models \sim(A \wedge COND(T,\omega))$ does not hold, but $EIN(T,\omega) = \nu$.

In many cases the whole set $WAY(T)$ as search space can be restricted by the following theorem.

THEOREM 7.2.

1. $RET(T,\mathfrak{D},A) = \emptyset$ if $\mathfrak{D} \models \sim A$
2. $RET(B(T_o,T_1),\mathfrak{D},A) = RET(T_o,\mathfrak{D},A)$ if $\mathfrak{D} \models A \rightarrow \sim B$
3. $RET(B(T_o,T_1),\mathfrak{D},A) = RET(T_1,\mathfrak{D},A)$ if $\mathfrak{D} \models A \rightarrow B$
4. $RET(t(\nu_1:T_1,\ldots,\nu_m:T_m),\mathfrak{D},A) = \bigcup_{\mu \in M} RET(T_\mu,\mathfrak{D},A)$
   if $\mathfrak{D} \models A \rightarrow \bigvee_{\mu \in M} t=\nu_\mu$ and $M \subseteq \{1,\ldots,m\}$

PROOF

Use the definition of RET.

Finally, we will try to use a given theorem tester or proof

system for checking whether, for instance, $\mathfrak{D} \models \sim(A \wedge COND(T, \omega))$, $\mathfrak{D} \models \sim A$, $\mathfrak{D} \models A \longrightarrow \sim B$, $\mathfrak{D} \models A \longrightarrow B$ or not.

REFERENCES (selected from references of the full draft version)

[1] BURMEISTER, P.: A Model Theoretic Oriented Approach to Partial Algebras. Akademie-Verlag Berlin 1986, 319 pages

[2] CHEN, K. and Z. RAS: DDH-approach to information systems. Proceedings of the 982 CISS in Princeton, N.J., 521-526

[3] CHEN, K. and Z. RAS: Dynamic hierarchical data bases. Proceedings of the 1983 ICAA in Taipei, Taiwan

[4] CHEN, K. and Z. RAS: An axiomatic theory of information trees. The University of North Carolina at Charlotte. Department of Mathematics and Computer Science, 1984

[5] CHEN, K. and Z. RAS: Homogeneous information trees. Fundamenta Informaticae 8,1 (1985), 123-149

[6] JACOBS, B. E.: On database logic. Journal of the ACM 29,3 (April 1982), 310-332

[7] KNUTH, D. E.: The Art of Computer Programming. Vol. 3. Sorting and Searching, 1973

[8] LÜDDE, E.: Zur Optimierung fragebogentheoretisch definierter Suchverfahren. Dissertation A, Humboldt-Universität zu Berlin, 1976

[9] MAREK, W. and Z. PAWLAK: Mathematical foundations of information storage and retrieval I, II, III. CC PAS Reports, I: No. 135, II: No. 136, III: No. 137 (1973), Warsaw

[10] PAWLAK, Z.: Foundations of information retrieval. CC PAS Reports, No. 101, Warsaw 1973

[11] PICARD, C. F.: Theorie des questionnaires, Paris 1965
Graphes et questionnaires. Paris 1972

[12] RAS, Z: An algebraic approach to information retrieval systems, Int. Journal of Comp. and Inf. Sciences, Vol. 11, No. 4, (August 1982), 275-293

[13] SALTON, G.: Automatic Information Organization and Retrieval. New York 1968

[14] SANDEWALL, E.: New Computing Environment, Software System Research Center, Linköping University, Sweden, 1980

[15] THIELE, H.: Wissenschaftstheoretische Untersuchungen in algorithmischen Sprachen I. Theorie der Graphschemata-Kalküle. Berlin 1966

[16] THIELE, H.: On a graph-theoretic realization of retrieval systems. Colloques Internationaux du C.N.R.S., No. 276. THEORIE DE L'INFORMATION, Paris (1977), 417-428

[17] THIELE, H.: On equivalent transformations of binary search trees. "Cinquieme Colloque de Lille Les arbres en algebre et en programmation", Lille (1980), 95-109

Searching and retrieval in databases by trees

H. Thiele

Summary

The paper deals with a logical approach to searching and retrieval in relational databases by trees. The starting point is a many-sorted first-order language with functional symbols, predicate symbols and the symbol = for the identity. Then, using the notion of signature, databases or, in algebraic terms, heterogeneous /operational-relational/ algebraic systems are defined. Two kinds of semantics for trees are defined: trees working in identification mode and trees working in retrieval mode. For both of these semantics the notions of soundness, completenes and /partial/ equivalence are introduced and theorems of decidability and axiomatizability are formulated relating to these notions. The paper concludes with two theorems giving a basis for a /fast/ retrieval algorithm using a proof system.

Keresés és visszakeresés adatbázisokban fák segitségével

H. Thiele

Összefoglaló

A cikk a relációs adatbázisokra vonatkozó keresés és visszakeresés logikai megközelitését tárgyalja. A kiinduló pont egy több-szortu első-rendü nyelv funkcionális szimbólumokkal, predikátum szimbólumokkal és az egyenlőség szimbólummal. Használva a szignatura fogalmát, adatbázist ill. heterogén algebrai rendszereket lehet definiálni. A fákra vonatkozó kétfajta szemantikátdefiniál a szerző: identifikációs módban ill. visszakeresési módban müködő fák. Mindkét szemantikában a következő fogalmakat vezeti be a szerző: megalapozottság, teljesség és /parciális/ ekvivalencia. Ezekre a fogalmakra aztán eldönthetőségi és axiomatizálhatósági tételeket bizonyit. A cikk végén két tétel van, amelyek segitségével gyors visszakeresési algoritmusok adhatók meg.