COMPLEXITY OF ALGORITHMS THAT CONSTRUCT SUBSETS
OF DATA FROM FORMAL DEFINITIONS

R. M. Nuriev

Polynomial-time algorithms that construct subsets of data from formal logical definition are considered.

This paper, like [1, 2], considers formal systems of logical definitions of data subsets. The aim of these studies is to find more powerful subsetting tools for data used in concurrent processing in different stages of execution of parallel programs. These tools may be viewed as query organization tools in (primarily relational) databases, which naturally allow logical queries. However, systems of sets induced by functional dependencies between attribute sets are inaccessible to the user, nor do they always correspond to the actual data representation; finally, these subset generation tools are insufficiently flexible. A more natural approach is therefore to organize the data as systems of subsets. This is the standard organization of library catalogs, systems of economic indicators, journals, documents, etc.

The formalization of definitions proposed in [1] is basically intended for such a description. A higher expressive power (than OBI, say) is achieved not only by the use of recursive definitions, but also by the multilevel structure of the system of concepts [2, 3]. The experience with the OTCHET operating environment [3] based on this formalization shows that these features, as well as the possibility of adequate description of the application domain, provide powerful tools for formal specification of programs for processing of large volumes of data. The use of these tools as a programming language construct, however, is justified only if we can develop effective algorithms (running in not more than polynomial time) for generation of sets from their definitions. This is the topic of our paper.

The problem is nontrivial, because the definitions use formulas of the predicate calculus, whose satisfiability is a classical NP-problem. We show that polynomial-time algorithms can be developed by introducing set names, considering the structural features of the definitions, as well as some types of extra-logical information about the properties of interpretations and the features of the sets being defined, and using fuzzy quantifiers.

A complete formal description of the system of definitions of data subsets is given in [2]. Here we basically present an informal discussion.

BASIC CONCEPTS

Systems of definitions are finite systems of elementary definitions, called forms or definitions, which are classified into four categories:

a) $S_{C_0}^i = \{ C_1, \ldots, C_n; C_0 \in S_0 \}$ is the definition of sets with the names $S_{C_0}^i$ by enumeration of their elements $C_1, \ldots, C_n$ for each $C_0 \in S_0$;

b) $S_v^i = \{ \Phi(v); U \}$ is the definition of sets with the names $S_v^i$ by specifying the property $\Phi(v)$ of the set elements in combination with the elements of this or other sets; $\Phi(v)$ is a formula with a free variable $v$ and with set-bounded quantifiers whose names are listed in $U$, i.e., $U$ is a list of expressions of the form $x \in S_w^j$ with the subscripts $w$ in $U$ described by expressions of the form $w \in S_{x_k}^k$;

c) $S_v^i = \{ S_{t(v)}; \Phi(\mathcal{X}_1, y, v); U \}$ is inductive definition of sets with the names $S_v^i$; $S_{t(v)}$ is the induction basis, $\Phi(\mathcal{X}_1, y, v)$ is the rule for deducing the membership of $y$ in the set $S_v^i$ from the membership of $\mathcal{X}_1$ in this set; this rule is a formula in the predicate calculus with bounds on the quantifiers in $U$;

d) $S_v^i = \{ t(\mathcal{X}_1, \ldots, \mathcal{X}_n); \Phi; U \}$ is the definition of the elements of sets with the names $S_v^i$ as images of the elements $\mathcal{X}_1, \ldots, \mathcal{X}_n$ under the mapping $t$; the choice of the elements $\mathcal{X}_1, \ldots, \mathcal{X}_n$ from sets listed in $U$ is determined by formula $\Phi$.

The superscript \( i \) denotes the set type.

Elements of sets whose representatives are the variables \( x_1, \ldots, x_n, v, y \) may be data from sets of type \( a \) and \( b \) and set names. In the latter case, the variables \( x_i \) have the form \( F_{yi} \); they indicate either membership of \( y_i \) in some set or membership in a type in the form \( S \in S^i \).

The set of forms \( F \) is called an open system with respect to \( S \) if the set of all types used in the right-hand sides of the definitions from \( F \) includes the set \( S \), and for each type not included in \( S \) there is one and only one form in \( F \) that defines this type. In what follows, we use the notation \( D(F) \) for the set \( S \) and \( O(F) \) for all other set types. A system is called closed if \( O(F) = \emptyset \).

Let \( F \) be a system, \( G \) a family of (named) set pairs \([\text{set name, enumeration of set elements}]\) such that the types from \( D(F) \) are contained in the set of types from \( G \). Then we say that the family of named sets \( M \) is extracted by the system \( F \) for the given \( G \) and a given interpretation of function and predicate symbols in \( F \) if the sets from \( G \cup M \) satisfy each definition from \( F \) and any other family that differs from \( M \) by one of the sets does not satisfy at least one definition in \( F \).

Several families may satisfy the same definition. Each of these families is called an extraction variant.

The problem of constructing sets from definitions is considered in the class of systems without parametric names and for subclasses of systems of forms with an increasing complexity, where complexity is defined as the number of set types, the number of quantifiers, and the number of quantifier changes. Subclasses are defined by integer vectors of variable length, in which the first component is the number of set types, the second component is the maximum number of quantifiers in various definitions, the third component is the number of quantifier changes, and so on. A vector of length \( k \) contains the first \( k \) characteristics. For instance, the class \((1, 1, 1)\), with which we start our discussion, indicates systems of definitions with one set type and one quantifier. This class includes systems of the form

\[
  S = \{v_1, \ldots, v_n\};
\]

\[
  S = \{\forall x \in S \exists y \in S p(x, y)\};
\]

\[
  S = \{\exists x \in S \forall y \in S q(x, y)\};
\]

The algorithm for constructing the corresponding sets is obvious.

The subclass \((1, 2)\) contains systems of the following four categories:

1) \( S = \{\forall x \in S \forall y \in S p(x, y)\}; \)
2) \( S = \{\forall x \in S \exists y \in S p(x, y)\}; \)
3) \( S = \{\exists x \in S \forall y \in S p(x, y)\}; \)
4) \( S = \{\exists x \in S \exists y \in S p(x, y)\}. \)

**Lemma 1.** For any system from the subclass \((1, 2)\), a polynomial time algorithm exists for constructing the corresponding sets.

**Proof.** For a system of category 1), if \( a \in M(S) \), then \( p(a, a) = T \), because an admissible one-element set always exists if \( M(S) \neq \emptyset \).

If \( V \) contains an element \( v \) such that for any \( x \in M(S) \) we have \( p(x, v) \) \& \( p(v, x) \) \& \( p(v, v) = T \), then the admissible set can be augmented with this element. An admissible set that cannot be augmented with any element from \( V \setminus M(S) \) is the sought extraction variant.

The algorithm is obviously correct by construction.

Remark 1. The algorithm can be generalized to a greater number of universal quantifiers. Then the first element \( a \) is chosen from the condition \( p(a,a,\ldots,a) = T \) and the extension in step \( k + 1 \) is based on the condition \( \forall y_1 \in M_k \forall y_2 \in M_k \ldots \forall y_1 \in M_k \exists v \exists y_2 \in M_k \ldots \exists y_1 \in M_k \left[ p(v, y_2, \ldots, y_1) \& p(y_1, v, y_2, \ldots, y_1) \& \ldots \& p(y_1, \ldots, y_1, v) \& p(y_1, \ldots, y_1, v) \right]. \)

For systems of category 2), the algorithm identifies in a stepwise manner sets of data that are not contained in \( S \).

In the first step, it eliminates the elements \( x \in V \) for which \( p(x, y) = F \) for each \( y \in V \). Note that if there are no such \( x \), then \( V \) is the sought set.

In the second and each successive step \( k \), the algorithm eliminates the elements with the same property from the universe \( V \setminus \{X_1 \cup \ldots \cup X_{k-1}\} \), where \( X_i \) are the elements removed in stage \( i \). For \( k = 2 \), we have \( V = V \setminus X_1 \).

The process ends either when we have eliminated all the elements from the universe or when no elements are rejected in some stage.