INFORMATICA, 1990, Vol.1, No.2, 110–120

STRUCTURAL SYNTHESIS OF DATA PROCESSING PROGRAMS

Justinas LAURINSKAS and Gražina TAUČAITĖ

Institute of Mathematics and Informatics,
Lithuanian Academy of Sciences,
232600 Vilnius, K.Požėlos St.54, Lithuania

Abstract. Data processing programs combine computations with navigation in a data base. Methods of the structural synthesis of programs are oriented towards construction of computing programs, and the methods of synthesis of relations are oriented towards construction of (logical) navigation paths. An approach to the integration of methods of the structural synthesis of programs and methods of the synthesis of relations is proposed. This approach is based on a computation model (a set of formulas), describing both computations and navigation paths. The sound and complete system of inference rules for the class of formulas, used in such computation models, is given.

Key words: structural synthesis of programs, relational data bases, synthesis of relations, inference rules.

Introduction. The structural synthesis of programs (Tyugu and Harf, 1980; Dikovskij and Kanovich, 1985) is based on a specification of computations called a computation model. A computation model describes the relationships among the initial program modules (operations) and is a set of named initial dependencies among some variables (values). A name of an initial dependency is a name of the program

module which implements that dependency. The required program is specified with the help of two sets: the set of given data (variables) and the set of desired data (variables), i.e. with the help of unnamed dependency. The structural synthesis consists in the search for the corresponding named dependency, which is logically implied by the initial dependencies. The name of this derived dependency represents the plan of a program (in the form of a composition of initial modules), which implements that dependency.

The synthesis of relations (Lozinskii, 1980; Honeyman, 1980) is based on a specification of a relational database called a database scheme. A database scheme describes base relations and is a pair of sets: the set of named dependencies, representing the schemes of base relations, and the set of (unnamed) dependencies, describing the properties of base relations. A relation scheme is a set of attributes that define the format of the relation. In addition, the relational operations enabling to obtain new relations from the base ones are defined. The required relation is specified with the help of its scheme, i.e. with the help of unnamed dependency. The synthesis of the relation consists in the search for the corresponding derived dependecy, which is logically implied by a database scheme and which name represents the relational expression that defines the required relation in terms of base relations.

The structural synthesis of programs does not take into account the possibilities of implementation of some initial dependencies by means of relations and does not consider possibilities of computation of relations by means of relational operations. On the other hand, the synthesis of relations does not consider possibilities of computation of relations by means of arbitrary (unrelational) operations. In this paper an approach to the integration of the structural synthesis of programs and the synthesis of relations is proposed, i.e. an approach to the

structural synthesis of data processing programs, which combine computations by means of arbitrary operations with the computations by means of relational operations.

The usual method to describe a set of logically implied dependencies is the method of formal theories. An inference rule of a theory indicates how to derive dependency from other dependencies. If the theory (the set of inference rules) is sound and complete, then the conditions "a dependency is logically implied from a set of dependencies" and "a dependency is derived from a set of dependencies" are equivalent.

A computation model (the database scheme) is defined over a universal set of variables (attributes) U and a set of functionals G. Each variable is associated with the set called its domain. A set of variables is some subset of the set U. The set G is the union of sets $G_0, G_1, \ldots$, where G_i is a set of functionals of rank i. A functional of rank i corresponds to a program module with i parameters–subprograms. The functionals with zero rank are called functions. We assume that the set of functions contains an empty (identity) function T.

Let us agree, for the union of sets of variables X and Y, to use notation XY alongside with the usual notation, and for the set of variables $\{A_1, A_2, \ldots, A_k\}$ notation $A_1A_2 \ldots A_k$.

To simplify discussion we do not consider conditional dependencies here.

Structural synthesis of programs. An expression of the form $X \to Y$ is called functional dependency (F–dependency), and an expression of the form

$$(X_1 \to Y_1)\& \ldots \&(X_k \to Y_k) \to Z$$

is called operational dependency (O–dependency). Here X, Y, X_i, Y_i, and Z are sets of variables. The number $k > 1$ indicates the rank of an O–dependency (the rank of an F–dependency equals 0).

A named dependency is an expression of the form $P : d$, where P is a name of a dependency, and d is a dependency. A name of a dependency is a functional or a program plan.

An initial dependency is a named dependency of the form $F : d$, where F is a functional, which rank coincides with that of the dependency d. The initial dependency $F : X \to Y$ expresses the computability of Y from X by means of module F. The initial dependency $F : (X_1 \to Y_1)\& \ldots \&(X_k \to Y_k) \to \to Z$ expresses the computability of Z by means of module F in the presence of computability of Y_i from X_i, $1 \leqslant i \leqslant k$.

A derived dependency is a named dependency of the form $P : d$, where P is a program plan, and d is an F–dependency. The derived dependency $P : X \to Y$ expresses the computability of Y from X by means of the program P. The set of program plans is defined as follows: if F is a function, then F is a program plan; if F is a functional of rank $k > 1$ and $P_1, \ldots, P_k$ are program plans, then $F(P_1, \ldots, P_k)$ is a program plan; if P_1 and P_2 are program plans, then $(P_1; P_2)$ is a program plan (a composite sequential operator).

The model-theoretic semantics of dependencies is based on the notion of state as a mapping from U to $\{0, 1\}$. If t is a state, then the condition $t[A] = 1$ ($t[A]$ denotes the value of t on $A \in U$) means that the value A is computable in t, and the condition $t[A] = 0$ means that the value A is not computable in t. The state t satisfies: $A \in U$ if $t[A] = 1$; $X \subseteq U$ if t satisfies all $A \in X$; $X \to Y$ if whenever t satisfies X it also satisfies Y; $(X_1 \to Y_1)\& \ldots \&(X_k \to Y_k) \to Z$ if whenever t satisfies all $X_i \to Y_i$ it also satisfies Z.In a similar way, by extending the notion of state as a mapping from $U \cup G$ to $\{0, 1\}$, it is also possible to define the semantics of named dependencies.

A task of the structural synthesis of programs is a triple (D, X, Y), where D is a set of initial dependencies and X and Y are sets of variables. The solution of the task (D, X, Y) is

the program plan P, such that the dependency $P : X \to Y$ is logically implied by a set D.

The following set of inference rules is sound and complete for the derivation of F–dependencies from a set of F– and O–dependencies:

$S1$. $T : X \to X$.

$S2$. $P : X \to Y$ implies $\mathrm{P} : \mathrm{X} \to \mathrm{Z}$ if $\mathrm{Z} \subseteq \mathrm{Y}$.

$S3$. $P_1 : X \to Y$ and $\mathrm{P}_2 : \mathrm{Y} \to \mathrm{Z}$ imply $(\mathrm{P}_1; \mathrm{P}_2) : \mathrm{X} \to \mathrm{YZ}$.

$S4$. $P_1 : WX_1 \to Y_1, \ldots, P_k : WX_k \to Y_k$ and
$F : (X_1 \to Y_1)\& \ldots \&(X_k \to Y_k) \to Z$ imply
$F(P_1, \ldots, P_k) : W \to Z$.

Example 1. As an example, let us describe a fragment of the salary computation model $SALARY1$ (the example is taken from Dikovskij and Kanovich (1985)):

F_1 :$EMP.YEAR \to REC$,
F_2 :$EMP.YEAR.MONTH \to SAL$,
F_3 :$REC.A\_SAL \to T\_SAL$,
F_4 :$(YEAR.MONTH \to SAL)\&(\to YEAR) \to A\_SAL$.

In this model EMP stands for employee name, REC for record, SAL for month's salary, A_SAL for average month's salary, and T_SAL for "thirteenth" salary. In the model $SALARY1$ it is possible, for example, to formulate a task $TASK1$: "for employee $EMP = e$ find the 13th salary obtained per year $YEAR = y$", i.e. the task ($SALARY1$, $EMP.YEAR$, T_SAL). This task is solvable: F_2 and F_4 imply $F_4(F_2)$: $EMP.YEAR \to A\_SAL$; F_1 and F_3 imply $(F_1; F_3)$: $EMP.YEAR.A\_SAL \to T\_SAL$; $F_4(F_2)$ and $(F_1; F_3)$ imply the named dependency $(F_4(F_2); F_1; F_3)$:

$EMP.YEAR \rightarrow T\_SAL$. However, in the model $SALARY1$ it is impossible, for example, to formulate a task $TASK2$: "for all employees whose work record in the year $YEAR = y$ has not exceeded the value $REC = r$ find employee names and their 13th salaries obtained per year $YEAR = y$". A problem with the models of the given type is that they do not take into account that some dependencies (dependencies F_1 and F_2 of our example) are represented by their extentionals.

Relation synthesis. A tuple on set of attributes X is a function t mapping each attribute $A \in X$ to an element in its domain. If t is a tuple on X and Y is a subset of X, then $t[Y]$ denotes a restriction of t on Y. A relation s with the scheme S, $s(S)$, is a set of tuples on S.

A relation $s(S)$ satisfies the F–dependency $X \rightarrow Y$ if XY is a subset of S and for all tuples t_1 and t_2 in s $t_1[X] = t_2[X]$ implies $t_1[Y] = t_2[Y]$.

Given a relation $s(S)$ and a set of attributes $X \subseteq S$, the projection of $s(S)$ onto X, denoted $\pi_X(s)$, is a relation with the scheme X, containing all the tuples $t[X]$ such that t belongs to s. The join of relations $s_1(S_1)$ and $s_2(S_2)$, denoted $s_1 * s_2$, is a relation with the scheme S_1S_2, containing all the tuples t on S_1S_2 such that $t[S_1]$ belongs to s_1, and $t[S_2]$ to s_2.

A database scheme is a pair (R, D), where $R = \{R_1 : S_1, \ldots, R_n : S_n\}$ is a set of named schemes of base relations, such that $S_1 \ldots S_n = U$, and D is a set of F–dependencies. A database state r is a set of base relations, $r = \{r_1(S_1), \ldots, r_n(S_n)\}$.

A relation $u(U)$ is called a universal relation for the database state r if u satisfies D and every r_i is the projection of u onto S_i.

A relational expression over a database is a relational expression, which operands are the names of base relation schemes. If E is a relational expression over a database, then

$E(r)$ denotes the value of E when each operand R_i of E is replaced by r_i.

We say expression of the form $E : S$ is an entity dependency (E–dependency) if E is a relational expression over a database and S is the scheme of E (the scheme of the resulting relation). Let $E_1 : S_1$ and $E_2 : S_2$ be two E–dependencies. We say $E_1 : S_1$ is a subdependency of $E_2 : S_2$ if E_1 is a subexpression of E_2. Let u be a universal relation for r. We say r satisfies E–dependency $E : S$ if $E(r) = \pi_S(u)$ and r satisfies all subdependencies of $E : S$.

F–dependencies describe the properties of the database state, and E–dependencies express the computability of relations by means of relational operations.

A task of the relation synthesis is a pair $(R \cup D, X)$, where (R, D) is a database scheme and X is a set of attributes. The solution of the task $(R \cup D, X)$ is a relational expression E such that the dependency $E : X$ is logically implied by a set $R \cup D$.

The following set of inference rules is sound and complete for the derivation of E–dependencies from a set of E–and F–dependencies:

$R1$. $X \to X$.

$R2$. $X \to Y$ implies $X \to Z$ if $Z \subseteq Y$.

$R3$. $X \to Y$ and $Y \to Z$ imply $X \to YZ$.

$R4$. $E : S$ implies $\pi_X(E) : X$ if $X \subseteq S$.

$R5$. $E_1 : S_1, E_2 : S_2$, and $S_1 \cap S_2 \to S_1$ or $S_1 \cap S_2 \to S_2$ imply $(E_1 * E_2) : S_1 S_2$.

Example 2. Let us describe the database scheme $SALARY2$:

$$R_1 : EMP.YEAR.REC, EMP.YEAR \to REC,$$

$R_2 : EMP.YEAR.MONTH.SAL,$
$EMP.YEAR.MONTH \rightarrow SAL,$
$R_3 : EMP.YEAR.A\_SAL, EMP.YEAR \rightarrow A\_SAL,$
$R_4 : EMP.YEAR.T\_SAL, EMP.YEAR \rightarrow T\_SAL.$

In the database scheme $SALARY2$ there may be formulated and solved the relation synthesis task $(SALARY2, EMP.YEAR.REC.T\_SAL) : R_1, R_4$, and $EMP.YEAR \rightarrow \rightarrow T\_SAL$ imply $(R_1 * R_4) : EMP.YEAR.REC.T\_SAL$. It is obvious that the relation $R_1 * R_4$ may be used when solving the task $TASK2$ (see Example 1). However, the database scheme $SALARY2$ does not take into account that relations R_3 and R_4 are not independent and that they are computed on the basis of relations R_1 and R_2.

Structural synthesis of data processing programs. A subset of a relation $s(S)$, containing all the tuples with identical restrictions on X, is called an X–cut of $s(S)$, where $X \subseteq S$.

Let us consider the O–dependency $F : (X_1 \rightarrow Y_1)\& \ldots \ldots \& (X_k \rightarrow Y_k) \rightarrow Z$ and the relation $s(S)$. Let $X_1Y_1 \ldots \ldots X_kY_kZ$ is a subset of S and there exist k sets of attributes W_i such that $W_i \cap Y_i = \varnothing$ whenever $X_i \neq \varnothing$, $1 \leqslant i \leqslant k$. We say $s(S)$ satisfies the O–dependency F if whenever $s(S)$ satisfies all $W_iX_i \rightarrow Y_i$ it also satisfies $W \rightarrow Z$ and

$$\pi_{WZ}(s) = F(\pi_{W_1X_1Y_1}(s), \ldots, \pi_{W_kX_kY_k}(s)),$$

where $W = W_1 \ldots W_k$.

A database scheme with computations is a pair (R, D), where $R = \{R_1 : S_1, \ldots, R_n : S_n\}$ is a set of named schemes of base relations (a set of E–dependencies) such that $S_1 \ldots S_n$ is a subset of U and D is a set of F–and named O–dependencies.

Let us extend the concept of E –dependency: as an E–dependency we shall comprehend an expression of the form

$P : S$, where P is the plan of a data processing program and S (as before) is the relation scheme. The set of plans of data processing programs is determined as follows: if E is a relational expression over database, then E is the plan of a data processing program; if F is a functional of rank k and $P_1, \ldots, P_k$ are the plans of data processing programs, then $F(P_1, \ldots, P_k)$ is the plan of a data processing program.

F–dependencies describe the properties of the database state, O–dependencies express the computability of relations by means of unrelational operations, and E–dependencies express the computability of relations by means of unrelational and relational operations.

A pair $(R \cup D, X)$ is a task of the structural synthesis of data processing programs, where (R, D) is the database scheme with computations and X is a set of attributes. A solution of the task $(R \cup D, X)$ is the plan of data processing program P such that the dependency $P : X$ is logically implied by a set $R \cup D$.

The following set of inference rules is sound and complete for the derivation of E–dependencies from a set of $F-$, O–and E–dependencies:

$P1$. $X \to X$.

$P2$. $X \to Y$ implies $X \to Z$ if $Z \subseteq Y$.

$P3$. $X \to Y$ and $Y \to Z$ imply $X \to YZ$.

$P4$. $W_1X_1 \to Y_1, \ldots, W_kX_k \to Y_k$,
and $F : (X_1 \to Y_1)\& \ldots \&(X_k \to Y_k) \to Z$
imply $W_1 \ldots W_k \to Z$ if $W_i \cap Y_i = \varnothing$
whenever $X_i \neq \varnothing$, $1 \leqslant i \leqslant k$.

$P5$. $E : S$ implies $\pi_X(E) : X$ if $X \subseteq S$.

$P6$. $E_1 : S_1$, $E_2 : S_2$, and $S_1 \cap S_2 \to S_1$
or $S_1 \cap S_2 \to S_2$ imply $(E_1 * E_2) : S_1S_2$.

P7. $E_1 : S_1, \ldots, E_k : S_k$,
$W_1X_1 \to Y_1, \ldots, W_kX_k \to Y_k$,
$W_1X_1Y_1 = S_1, \ldots, W_kX_kY_k = S_k$,
and $F : (X_1 \to Y_1)\& \ldots \&(X_k \to Y_k) \to Z$ imply
$F(E_1, \ldots, E_k) : W_1 \ldots W_kZ$ if $W_i \cap Y_i = \varnothing$
whenever $X_i \neq \varnothing$, $1 \leqslant i \leqslant k$.

Example 3. Let us describe the database scheme with computations $SALARY3$:

$$R_1 : EMP.YEAR.REC, EMP.YEAR \to REC,$$
$$R_2 : EMP.YEAR.MONTH.SAL,$$
$$EMP.YEAR.MONTH \to SAL,$$
$$F_1 : (\to REC)\&(\to A\_SAL) \to T\_SAL,$$
$$F_2 : (MONTH \to SAL) \to A\_SAL.$$

The task ($SALARY3$, $EMP.YEAR.REC.T\_SAL$) is solved as follows: R_2, F_2, and $EMP.YEAR.MONTH \to \to SAL$ imply $F_2(R_2) : EMP.YEAR.A\_SAL$ and $EMP.YEAR \to A\_SAL$; F_1, $F_2(R_2)$, R_1, $EMP.YEAR \to \to A\_SAL$, and $EMP.YEAR \to REC$ imply $F_1(R_1, F_2(R_2)) : EMP.YEAR.T\_SAL$ and $EMP.YEAR \to T\_SAL$; R_1, $F_1(R_1, F_2(R_2))$, and $EMP.YEAR \to T\_SAL$ imply $(R_1 * *F_1(R_1, F_2(R_2))) : EMP.YEAR.REC.T\_SAL$.

Conclusions

1. The methods of structural synthesis of programs do not take into account the possibilities of implementation of some initial dependencies by means of relations and do not consider the possibilities of computation of relations by means of relational operations.

2. The methods of relation synthesis do not consider a possibility of computation of relations by means of arbitrary (unrelational) operations.

3. The proposed approach to the program synthesis is based on a computation model (a database scheme with computations), describing computations by means of both unrelational and relational operations. A sound and complete system of inference rules for the class of dependencies, used in such computation models, is given.

REFERENCES

Dikovskij, A.Ja., and M.I.Kanovich (1985). Computation models with separable subtasks. *Izv. AN SSSR. Tekhn. kibernetika,* **5**, 36–59 (in Russian).

Honeyman, P. (1980). Extension joins. In *Proc. 6th Int. Conf. Very Large Data Bases (Montreal,Canada, Oct. 1-3, 1980). ACM,* New York. pp. 239–244.

Lozinskii, E.L. (1980). Construction of relations in relational databases. *ACM Trans. Database Syst.,* **5**(2), 208–224.

Tyugu, E.H., and M.Ja.Harf (1980). Algorithms of the structural synthesis of programs. *Programirovanije,* **4**, pp. 3–13 (in Russian).

Received January 1990

J. Laurinskas received the Degree of Candidate of Technical Sciences from the Kaunas Polytechnic Institute, Kaunas, Lithuania, in 1976. He is a senior researcher at the Department of Mathematical Logic and Theory of Algorithms of the Institute of Mathematics and Informatics of the Lithuanian Acad. Sci. He is engaged in the problems of designing of intellectual programming systems.

G. Taučaitė is a researcher at the Department of Mathematical Logic and Theory of Algorithms, Institute of Mathematics and Informatics, Lithuanian Acad. Sci. Her interests include the designing of intellectual programming systems.