Acta Informatica 4, 201—211 (1975)

Grundlagen einer Theorie der Datenstrukturen und Zugriffssysteme

Teil I: Datenstrukturen und Schemata

H.-D. Ehrich

Eingegangen am 8. Februar 1974

Summary. A mathematical model for data structures is presented from which other well known and widely used methods of representation can be derived easily. In this model the schema/instance relationship which is especially important with regard to data base systems, is described in terms of partitions, quotient structures, and homomorphisms of data structures. The main result is an algebraic characterization of the set of possible schemas for a given data structure.

1. Einleitung

Die theoretischen Ansätze und Grundlagenuntersuchungen, über die in dieser Arbeit berichtet wird, behandeln die Möglichkeiten, Datenstrukturen und schematische Beschreibungen von Datenstrukturen formal darzustellen.

Wesentliche Anregungen für die hier untersuchten Probleme kommen aus dem praktischen Problemkreis der Entwicklung allgemeiner Datenbanksysteme. In einem solchen System sollen die vom Benutzer formulierten Datenzugriffe organisatorisch möglichst getrennt werden von der aktuellen Speicherung der Daten. Das heißt, ein Benutzer soll sich nur formal — durch symbolische Namen oder Suchfragen — auf die Datenstruktur beziehen. Diese Unabhängigkeit der Datenstrukturen von der Implementierung ist eines der Hauptziele bei dem Entwurf von Datenbanksystemen (vgl. [3—5, 10, 11, 14, 24] und die Arbeiten in [20]). Nur so läßt sich erreichen, daß die Datenbank bei Bedarf erweitert und umorganisiert werden kann, ohne daß Benutzerprogramme geändert werden müssen. Für die Übersetzung der formal formulierten Benutzeranforderungen in E/A-Operationen ist es notwendig, daß das System zusätzlich zu den für die Benutzer relevanten Daten eine strukturelle Beschreibung dieser Daten, das „Schema", enthält [3, 11].

Im folgenden Abschnitt wird ein mathematisches Modell für Datenstrukturen definiert, und es wird erläutert, wie sich bekannte Darstellungsarten daraus herleiten lassen. Der Rest der Arbeit behandelt die strukturellen Beziehungen zwischen Datenstrukturen und ihren schematischen Beschreibungen im Modell. Hierzu werden Partitionen und Faktorstrukturen von Datenstrukturen eingeführt. Die wichtigsten Ergebnisse betreffen eine Charakterisierung schematischer Partitionen und einen Satz über die algebraische Struktur der schematischen Partitionen im Verband aller Partitionen einer Datenstruktur. Dann werden Homomorphismen von Datenstrukturen eingeführt, und es wird gezeigt, wie mit deren Hilfe ein adäquater Schema-Begriff definiert werden kann. Schließlich wird ge-

zeigt, daß die möglichen Schemata einer Datenstruktur im wesentlichen (bis auf Isomorphie) den schematischen Partitionen dieser Datenstruktur entsprechen, und daß es zu jeder Datenstruktur D ein (bis auf Isomorphie) eindeutig bestimmtes „feinstes" Schema S_f gibt mit der Eigenschaft, daß jedes Schema von D auch Schema von S_f ist.

2. Das mathematische Modell

Für die Formalisierung des Begriffes Datenstruktur wählen wir einen Ansatz, der bei Salton [21] in der Form des „relational data file" anklingt, und der explizit die Grundlage für die Beschreibung der Datenstrukturen in der assoziativen Sprache LEAP von Feldman und Rovner [12, 19] bildet. Andere Ansätze findet man in [2, 5—8, 13, 22—27].

Definition 2.1. Eine *Datenstruktur* ist ein Quadrupel $D=(\Omega, A, V, \varrho)$, wobei Ω, A und V endliche Mengen von Objekten, Attributen bzw. Werten sind. ϱ ist eine dreistellige Relation auf Ω, A und V: $\varrho \subset \Omega \times A \times V$. Folgende Bedingungen werden gefordert:

$$1.\ \mathrm{pr}_1(\varrho)=\Omega, \quad 2.\ \mathrm{pr}_2(\varrho)=A, \quad 3.\ \mathrm{pr}_3(\varrho)=V.$$

Hierbei ist $\mathrm{pr}_i(\varrho)$, $i=1, 2, 3$, die Projektion der i-ten Komponente von ϱ, d.h. die Menge aller Elemente, welche in der i-ten Komponente von ϱ vorkommen.

In diesem Modell wird eine Datenstruktur also durch eine homogene Menge von Tripeln (ω, a, v) beschrieben. Beziehungen zwischen verschiedenen Objekten werden dadurch dargestellt, daß Werte wiederum Objekte sein können, d.h., daß $\Omega \cap V \neq \emptyset$ ist. Dadurch ergibt sich ein enger Zusammenhang zu dem Modell des Datengraphen von Rosenberg [16—18]: wir fassen alle Tripel mit gleichem Attribut zu einer Menge zusammen und streichen die mittlere Komponente. Dadurch entsteht für jedes Attribut $a \in A$ genau eine zugeordnete zweistellige Relation

$$\lambda_a := \{(\omega, v) \in \Omega \times V \mid (\omega, a, v) \in \varrho\}.$$

Die Struktur

$$\Gamma(D) := (\Omega \cup V, \Lambda_a)$$

mit

$$\Lambda_a := \{\lambda_a \mid a \in A\}$$

ist dann eine Verallgemeinerung des Datengraphen von Rosenberg (dort sind die λ_a partielle Transformationen).

Faßt man alle Tripel mit gleichem Objekt zusammen, so erhält man eine Einteilung der Datenstruktur in „Sätze" (vgl. [13]).

Definition 2.2. Sei $D=(\Omega, A, V, \varrho)$ eine Datenstruktur und $\omega \in \Omega$. Der *Satz von* ω ist

$$R_\omega := \{(a, v) \mid (\omega, a, v) \in \varrho\}.$$

Durch diese Einteilung kommen wir zu der weithin gebräuchlichen Darstellung einer Datenstruktur als einer Menge von Sätzen, wobei in einem Satz alle „Eigenschaften" ((a, v)-Paare) eines Objektes zusammengefaßt sind. Komplexere Satzstrukturen erhält man, wenn Werte im Satz wiederum Objekte sind, welche durch ihren Satz ersetzt werden usw.

Eine weitere Darstellung von Datenstrukturen ist das Relationen-Modell von Codd [5, 6]. Auf der Grundlage dieses Modells sind verallgemeinerte Datenbanksysteme entworfen worden [1, 15]. Der formale Zusammenhang zwischen unserem Modell und dem Relationen-Modell ist nicht ganz so offensichtlich wie in den Fällen des Datengraphen und der satzorientierten Darstellung. In [9] wird gezeigt, daß die Datenstrukturen des Relationen-Modells (in der ersten Normalform) im wesentlichen mit den *einfachen* Datenstrukturen unseres Modells korrespondieren, welche folgendermaßen definiert sind:

Definition 2.3. Eine Datenstruktur $D=(\Omega, A, V, \varrho)$ heißt *einfach*, wenn ϱ eine partielle Abbildung der Form

$$\varrho: \Omega \times A \to V$$

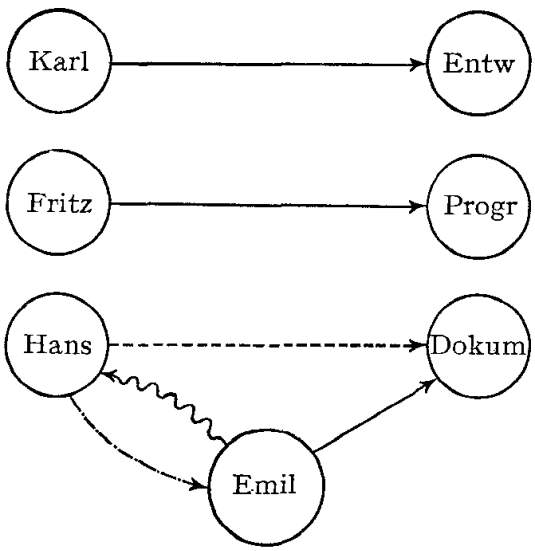
ist.

Ohne an dieser Stelle auf die Einzelheiten einzugehen, wollen wir an einem Beispiel die verschiedenen Darstellungsarten für Datenstrukturen demonstrieren.

Sei D_1 gegeben durch die folgende Menge von Tripeln:

(Karl, arbeitet an, Entwurf)
(Fritz, arbeitet an, Programmierung)
(Hans, leitet, Dokumentation)
(Emil, arbeitet an, Dokumentation)
(Emil, arbeitet unter, Hans)
(Hans, hat als Mitarbeiter, Emil).

Der zugehörige Datengraph läßt sich folgendermaßen darstellen:



——→: arbeitet an / - - - - →: leitet / ~~~~→: arbeitet unter
·····→: hat als Mitarbeiter

In der satzorientierten Darstellung würde der Satz des Objektes „Emil" folgendermaßen aussehen:

Emil: (arbeitet an: Dokumentation,
arbeitet unter: Hans)

oder bei Ersetzung von Hans durch dessen Satz:

Emil: (arbeitet an: Dokumentation,
arbeitet unter: *Hans:* (leitet: Dokumentation,
hat als Mitarbeiter: Emil)).

Dadurch entstehen hierarchische Baumstrukturen, deren Blätter jedoch gegebenenfalls miteinander oder mit anderen Knoten zu identifizieren sind. (Dies wird in der Praxis durch Verwendung von Verweisen als Werten erreicht.)

Im Relationenmodell schließlich würde man den obigen Teil einer Datenstruktur etwa durch eine n-stellige Relation mit dem Namen PERSONAL repräsentieren, wobei jedem Objekt ein n-Tupel (eine Zeile) und jedem Attribut eine Komponente der Relation (eine Spalte) zugeordnet ist:

	PERSONAL					
	Name	Nr.	arbeitet an	arbeitet unter	leitet	hat als Mitarbeiter
(Karl)	.....	.....	Entw.	.....	.....	.....
(Fritz)	.....	.....	Progr.	.....	.....	.....
(Hans)	.....	.....	.....	.....	Dokum.	Emil
(Emil)	.....	.....	Dokum.	Hans	.....	.....

3. Partitionen und Faktorstrukturen

Mit Hilfe unseres Modells sind wir in der Lage, Datenstrukturen auf eine konzeptionell einfache Art und Weise zu beschreiben, nämlich als homogene Menge von Tripeln; und wir haben gezeigt, wie unser Konzept mit anderen Arten der Darstellung zusammenhängt. Es handelt sich dabei in jedem Fall um die Beschreibung einer speziellen konkreten Datenstruktur, eines sog. „instance".

Um eine zentrale Verwaltung aller Daten in einer Datenbank durch ein Datenbanksystem zu ermöglichen, genügt es nicht, lediglich die für den Benutzer relevanten konkreten Datenstrukturen zu speichern. Bevor einem Benutzer erlaubt wird, Daten einzugeben, verlangt das System von ihm, daß er Art und Typ der Strukturen, die er verwenden will, deklariert. Der Benutzer muß also gewissermaßen eine „Grobstruktur" seiner Daten vorwegliefern, und seine aktuellen Datenstrukturen müssen nachher zu dieser Grobstruktur „passen". Eine solche Grobstruktur heißt „Schema" [3, 11], und das Datenbanksystem muß neben den aktuellen Daten auch Beschreibungen dieser Schemata enthalten.

Wir stellen uns hier auf den Standpunkt, daß Schemata auch Datenstrukturen sind und somit in unserem Modell beschrieben werden können. Dabei erhebt sich die Frage, welche strukturellen Beziehungen zwischen einem Schema und einem zugehörigen Instance bestehen.

Betrachten wir ein Beispiel. Die Datenstruktur im Beispiel des vorigen Abschnitts heiße D_1. D_2 sei die folgende Datenstruktur:

(Person A, arbeitet an, Projekt)
(Person A, leitet, Projekt)
(Person A, arbeitet unter, Person B)
(Person A, hat als Mitarbeiter, Person B).

Es ist klar, daß D_2 in gewissem Sinne eine vergröbernde schematische Beschreibung der Datenstruktur D_1 ist. Dies wird dadurch erreicht, daß die Objekte und Werte von D_1 zu Objekt- bzw. Wertetypen zusammengefaßt werden, welche dann die Objekte und Werte von D_2 bilden. Diese Grundidee wird im folgenden verallgemeinert und formalisiert.

Dazu benötigen wir einige Begriffe aus der Theorie der Partitionen einer Menge. Sei $X=\{x_1, \ldots, x_n\}$ eine endliche Menge. Eine Partition π von X ist eine Menge von Teilmengen von X, $\pi=\{X_1, \ldots, X_m\}$, so daß gilt:

1. $X_i \subset X$ für alle $i=1, \ldots, m$,
2. $X=\bigcup_{i=1}^{m} X_i$,
3. $X_i \cap X_j=\emptyset$ für $1 \leqq i<j \leqq m$.

Sei Π die Menge der Partitionen von X. Auf Π führen wir eine Relation $\leqq$ ein, die für $\pi=\{X_1, \ldots, X_m\} \in \Pi$ und $\tau=\{X'_1, \ldots, X'_p\} \in \Pi$ folgendermaßen definiert ist:

$$\pi \leqq \tau :\Leftrightarrow \forall i=1, \ldots, m\ \exists j=1, \ldots, p: X_i \subset X'_j.$$

Die Relation $\leqq$ ist reflexiv, transitiv und identitiv, also eine Halbordnungsrelation. Die Halbordnung $(\Pi, \leqq)$ ist ein Verband. Das Einselement ist die Einspartition

$$1 := \{X\},$$

und das Nullelement ist die Nullpartition

$$0 := \{\{x_1\}, \{x_2\}, \ldots, \{x_n\}\}.$$

Die beiden zweistelligen Verknüpfungen des Verbandes der Partitionen einer Menge X lassen sich am besten durch die zugehörigen Äquivalenzrelationen erklären. Seien R_1, R_2 Äquivalenzrelationen und $\pi(R_i)$, $i=1, 2$, die zugehörigen Partitionen auf X. Dann ist

$$\pi(R_1) \cdot \pi(R_2) := \pi(R_1 \cap R_2),$$
$$\pi(R_1) + \pi(R_2) := \pi\left((R_1 \cup R_2)^t\right),$$

wobei der obere Index t die Bildung der transitiven Hülle bedeutet.

Mit diesen allgemeinen Grundbegriffen können wir nun definieren, was eine Partition einer Datenstruktur und was die Faktorstruktur einer Datenstruktur nach einer Partition ist. Dazu sei $D=(\Omega, A, V, \varrho)$ eine Datenstruktur.

Definition 3.1. Eine *Partition* von D ist ein Tripel $\pi=(\pi_\omega, \pi_a, \pi_v)$, wobei π_ω, π_a und π_v Partitionen von Ω, A bzw. V sind.

Definition 3.2. Die Faktorstruktur D/π von D nach einer Partition $\pi=(\pi_\omega, \pi_a, \pi_v)$ von D ist die Datenstruktur

$$D/\pi := (\pi_\omega, \pi_a, \pi_v, \varrho/\pi),$$

wobei

$$\varrho/\pi := \{(B^\omega, B^a, B^v) \in \pi_\omega \times \pi_a \times \pi_v \mid \exists \omega \in B^\omega\ \exists a \in B^a\ \exists v \in B^v: (\omega, a, v) \in \varrho\}.$$

Die Faktorstruktur D/π ist durch Angabe der Datenstruktur D und der Partition π von D eindeutig bestimmt. Die Verbandsstruktur der Partitionen

einer Menge induziert auf einfache Weise eine Verbandsstruktur auf der Menge der Partitionen einer Datenstruktur (und damit auf der Menge der Faktorstrukturen einer Datenstruktur D nach den Partitionen von D), wenn man die Verknüpfung zwischen Partitionen $\pi=(\pi_\omega, \pi_a, \pi_v)$ und $\tau=(\tau_\omega, \tau_a, \tau_v)$ von D komponentenweise erklärt:

$$\pi \cdot \tau := (\pi_\omega \cdot \tau_\omega, \pi_a \cdot \tau_a, \pi_v \cdot \tau_v),$$
$$\pi + \tau := (\pi_\omega + \tau_\omega, \pi_a + \tau_a, \pi_v + \tau_v).$$

Für die zugehörige Halbordnungsrelation auf der Menge der Partitionen von D gilt dann:

$$\pi \leqq \tau :\Leftrightarrow \pi_\omega \leqq \tau_\omega \wedge \pi_a \leqq \tau_a \wedge \pi_v \leqq \tau_v.$$

Faktorstrukturen bilden ein adäquates Mittel, um die Beziehungen zwischen Datenstrukturen und den sie beschreibenden Datenstrukturen, den Schemata, zu untersuchen. So ist in dem Eingangsbeispiel D_2 eine Faktorstruktur von D_1 nach der Partition $\pi=(\pi_\omega, \pi_a, \pi_v)$, wobei gilt:

$$\begin{aligned}&\pi_\omega = 1,\\ &\pi_a = 0,\\ &\pi_v = \{\{\text{Entwicklung, Programmierung, Dokumentation}\},\\ &\qquad \{\text{Hans, Emil}\}\}.\end{aligned}$$

Eine andere Faktorstruktur von D_1 nach der Partition $\tau=(\tau_\omega, \tau_a, \tau_v)$ mit

$$\begin{aligned}&\tau_\omega = \{\{\text{Karl, Fritz, Emil}\}, \{\text{Hans}\}\},\\ &\tau_a = 0,\\ &\tau_v = \{\{\text{Entwicklung, Programmierung}\},\\ &\qquad \{\text{Dokumentation}\}, \{\text{Hans}\}, \{\text{Emil}\}\}\end{aligned}$$

wäre z.B. die folgende:

D_3: (Mitarbeiter A, arbeitet an, Projekt A)
(Leiter, leitet, Projekt B)
(Mitarbeiter A, arbeitet an, Projekt B)
(Mitarbeiter A, arbeitet unter, Leiter)
(Leiter, hat als Mitarbeiter, Mitarbeiter B).

4. Schematische Partitionen

Die letztere Faktorstruktur scheint nicht ganz dem zu entsprechen, was viele, die auf diesem Gebiet praktisch arbeiten, intuitiv unter einem Schema verstehen. Obwohl dieser Begriff nirgends präzise definiert ist, scheint doch eine gewisse Übereinstimmung in dem Punkt zu herrschen, daß ein Schema eine Art Charakterisierung der vorkommenden Attribute bzw. Attributtypen ist, daß also jedes Attribut im Schema genau einmal vorkommen sollte.

Wir untersuchen daher die folgenden speziellen Partitionen und Faktorstrukturen.

Definition 4.1. Eine Partition $\pi=(\pi_\omega, \pi_a, \pi_v)$ einer Datenstruktur $D=(\Omega, A, V, \varrho)$ heißt *schematisch*, wenn gilt:

$$|\pi_a| = |\varrho/\pi|.$$

Eine Faktorstruktur D/π heißt schematisch, wenn π schematisch ist.

Die Faktorstruktur einer Datenstruktur D nach der Nullpartition $0=(0, 0, 0)$ unterscheidet sich von D offenbar nur durch eineindeutige Bezeichnungsänderungen, ist also isomorph zu D (s.u.). Wir nennen daher eine *Datenstruktur* schematisch, wenn die Nullpartition von D schematisch ist, wenn also $|A|=|\varrho|$ ist. Da aufgrund der Definition einer Datenstruktur jedes Attribut in mindestens einem Tripel vorkommt, ist immer $|A|\leqq|\varrho|$. Gleichheit besteht daher genau dann, wenn jedes Attribut in genau einem Tripel vorkommt.

Zur Untersuchung der algebraischen Struktur der schematischen Partitionen im Verband aller Partitionen einer Datenstruktur D benötigen wir zunächst eine andere Charakterisierung der schematischen Partitionen.

Sei $D=(\Omega, A, V, \varrho)$ eine Datenstruktur, und sei $\pi=(\pi_\omega, \pi_a, \pi_v)$ eine Partition von D. Sei ferner $\pi_a=\{B_1^a, \ldots, B_r^a\}$. Wir ordnen jedem Block $B_i^a\in\pi_a$ die beiden Mengen

$$\beta(B_i^a):=\{\omega\in\Omega\mid\exists a\in B_i^a\,\exists v\in V\colon(\omega, a, v)\in\varrho\}$$

$$\gamma(B_i^a):=\{v\in V\mid\exists\omega\in\Omega\,\exists a\in B_i^a\colon(\omega, a, v)\in\varrho\}$$

von Objekten bzw. Werten zu. Aus den beiden Mengen

$$\{\beta(B_1^a), \ldots, \beta(B_r^a)\}$$

und

$$\{\gamma(B_1^a), \ldots, \gamma(B_r^a)\}$$

bilden wir zwei Partitionen $\sigma_\omega(\pi_a)$ auf Ω (bzw. $\sigma_v(\pi_a)$ auf V), indem wir die feinsten Partitionen bilden, für die gilt, daß jede Menge $\beta(B_i^a)$ (bzw. $\gamma(B_i^a)$) in einem der Blöcke von $\sigma_\omega(\pi_a)$ (bzw. $\sigma_v(\pi_a)$) enthalten ist.

Lemma. 4.1 Eine Partition $\pi=(\pi_\omega, \pi_a, \pi_v)$ einer Datenstruktur D ist genau dann schematisch, wenn sowohl $\pi_\omega\geqq\sigma_\omega(\pi_a)$ als auch $\pi_v\geqq\sigma_v(\pi_a)$ ist.

Beweis. Seien $\pi=(\pi_\omega, \pi_a, \pi_v)$ eine Partition von D, $\pi_a=\{B_1^a, \ldots, B_r^a\}$, und sei $\pi_\omega\geqq\sigma_\omega(\pi_a)$ sowie $\pi_v\geqq\sigma_v(\pi_a)$. Dann gilt für alle $i\in\{1, \ldots, r\}$, daß $\beta(B_i^a)\subset B^\omega$ und $\gamma(B_i^a)\subset B^v$ ist für je ein $B^\omega\in\pi_\omega$ und ein $B^a\in\pi_v$. Für jedes Attribut $a\in B_i^a$ gibt es ein Objekt $\omega\in\Omega$ und einen Wert $v\in V$, so daß $(\omega, a, v)\in\varrho$ ist, und es ist $\omega\in\beta(B_i^a)$ und $v\in\gamma(B_i^a)$. Dann ist auch $\omega\in B^\omega$ und $v\in B^v$, und wegen der Definition 3.2 ist (B^ω, B_i^a, B^v) das einzige Tripel aus ϱ/π mit B_i^a als zweiter Komponente. Da dies für alle $i=1, \ldots, r$ gilt, ist $|\varrho/\pi|=|\pi_a|$, also ist π schematisch.

Um die Umkehrung zu zeigen, sei $\pi_\omega\ngeqq\sigma_\omega(\pi_a)$ (für den Fall $\pi_v\ngeqq\sigma_v(\pi_a)$ gilt eine entsprechende Argumentation). Dann gibt es ein $i\in\{1, \ldots, r\}$ und zwei Objekte $\omega, \omega'\in\beta(B_i^a)$, so daß $\omega\in B^\omega\in\pi_\omega$ und $\omega'\in B^{\omega'}\in\pi_\omega$ ist, wobei $B^\omega\neq B^{\omega'}$. Nach der Definition von $\beta(B_i^a)$ gibt es Werte $v, v'\in V$ und Attribute $a, a'\in B_i^a$ mit (ω, a, v), $(\omega', a', v')\in\varrho$. Seien $v\in B^v\in\pi_v$ und $v'\in B^{v'}\in\pi_v$. Dann sind (B^ω, B_i^a, B^v) und $(B^{\omega'}, B_i^a, B^{v'})$ zwei verschiedene Tripel, die nach Definition 3.2 beide in ϱ/π sind. Dann muß aber $|\varrho/\pi|>|\pi_a|$ sein, π kann also nicht schematisch sein.

Wir halten nun eine Partition π_a auf der Attributmenge A fest und betrachten die Menge $\mathscr{P}(\pi_a)$ aller Partitionen einer Datenstruktur D mit identischer mittlerer Komponente:

$$\mathscr{P}(\pi_a)=\{\pi=(\pi_\omega, \pi_a, \pi_v)\mid\pi_a\text{ fest}\}.$$

Das Produkt $\pi \cdot \pi'$ und die Summe $\pi + \pi'$ zweier Partitionen $\pi, \pi' \in \mathscr{P}(\pi_a)$ liegt offenbar wieder in $\mathscr{P}(\pi_a)$, also ist $\mathscr{P}(\pi_a)$ ein Teilverband des Verbandes aller Partitionen von D. Der folgende Satz ergibt sich unmittelbar aus dem obigen Lemma und zeigt, daß die schematischen Partitionen, die in $\mathscr{P}(\pi_a)$ enthalten sind, einen Teilverband von $\mathscr{P}(\pi_a)$ bilden.

Satz 4.1. Seien $D = (\Omega, A, V, \varrho)$ eine Datenstruktur, π_a eine Partition von A und $\pi, \pi' \in \mathscr{P}(\pi_a)$ schematische Partitionen von D. Dann sind auch Produkt $\pi \cdot \pi'$ und Summe $\pi + \pi'$ schematische Partitionen aus $\mathscr{P}(\pi_a)$.

Beweis. Seien $\pi = (\pi_\omega, \pi_a, \pi_v)$ und $\pi' = (\pi'_\omega, \pi_a, \pi'_v)$. Da π und π' schematisch sind, gilt nach Lemma 4.1:

1. $\sigma_\omega(\pi_a) \leqq \pi_\omega$,
2. $\sigma_\omega(\pi_a) \leqq \pi'_\omega$,
3. $\sigma_v(\pi_a) \leqq \pi_v$,
4. $\sigma_v(\pi_a) \leqq \pi'_v$.

Aus 1. und 2. folgt $\sigma_\omega(\pi_a) \leqq \pi_\omega \cdot \pi'_\omega$, und aus 3. und 4. folgt entsprechend $\sigma_v(\pi_a) \leqq \pi_v \cdot \pi'_v$. Damit ist $\pi \cdot \pi'$ wieder schematisch. Wegen $\pi_\omega \leqq \pi_\omega + \pi'_\omega$ und $\pi_v \leqq \pi_v + \pi'_v$ folgt aus 1. und 3., daß auch die Summe $\pi + \pi'$ wieder eine schematische Partition ist.

Von besonderer Bedeutung für die Anwendung sind die „attributtreuen" schematischen Partitionen aus $\mathscr{P}(0_a)$, wobei 0_a die Nullpartition auf A ist. Die zugehörigen Faktorstrukturen geben eine eindeutige schematische Beschreibung aller in der Datenstruktur vorkommenden Attribute. Diese Faktorstrukturen nach attribut-treuen schematischen Partitionen hängen sehr eng mit einem anderen Datenstruktur-Modell zusammen, welches von Schwartz [22] benutzt worden ist, und sie bilden nach Satz 4.1 einen Teilverband von $\mathscr{P}(0_a)$.

5. Homomorphismen und Schemata

Faktorstrukturen nach schematischen Partitionen bilden noch keinen adäquaten Schema-Begriff, denn die Objekte, Attribute und Werte einer solchen Faktorstruktur D/π müßten sich bei dynamischer Erweiterung und Veränderung der Instance-Struktur D mit ändern, wenn die Schema-Instance-Beziehung erhalten bleiben soll. Um den Begriff des Schemas unabhängig von den speziellen Objekt-, Attribut- und Wertemengen der zu charakterisierenden Instance-Strukturen zu definieren, führen wir den Begriff des Homomorphismus ein.

Seien $D_i = (\Omega_i, A_i, V_i, \varrho_i)$, $i = 1, 2$, Datenstrukturen, und seien $\varphi_\omega: \Omega_1 \to \Omega_2$, $\varphi_a: A_1 \to A_2$, $\varphi_v: V_1 \to V_2$ Abbildungen. Wir fassen diese drei Abbildungen zu einem Tripel $\varphi = (\varphi_\omega, \varphi_a, \varphi_v)$ zusammen und nennen φ eine Abbildung von D_1 in D_2, in Zeichen $\varphi: D_1 \to D_2$.

Definition 5.1. Eine Abbildung $\varphi = (\varphi_\omega, \varphi_a, \varphi_v): D_1 \to D_2$ heißt *Homomorphismus* von D_1 in D_2 genau dann, wenn für alle $\omega \in \Omega_1$, $a \in A_1$ und $v \in V_1$ gilt:

(i) $(\omega, a, v) \in \varrho_1 \Rightarrow \big(\varphi_\omega(\omega), \varphi_a(a), \varphi_v(v)\big) \in \varrho_2$.

Ein Homomorphismus heißt *surjektiv* oder *von* D_1 *auf* D_2 genau dann, wenn zusätzlich gilt:

(ii) φ_ω, φ_a und φ_v sind surjektive Abbildungen,

(iii) $(\omega', a', v') \in \varrho_2 \Rightarrow \exists\, \omega \in \varphi_\omega^{-1}(\omega')\, \exists\, a \in \varphi_a^{-1}(a')$
$\exists\, v \in \varphi_v^{-1}(v') : (\omega, a, v) \in \varrho_1$.

φ heißt *Isomorphismus von* D_1 *in (auf)* D_2 genau dann, wenn φ ein Homomorphismus von D_1 in (auf) D_2 ist, und wenn zusätzlich gilt:

(iv) φ_ω, φ_a und φ_v sind injektive Abbildungen

D_1 und D_2 heißen *isomorph* genau dann, wenn es einen Isomorphismus von D_1 auf D_2 gibt.

Definition 5.2. Sie D eine Datenstruktur. Eine andere Datenstruktur S heißt *Schema* von D genau dann, wenn gilt:

(i) S ist schematisch,

(ii) Es gibt einen surjektiven Homomorphismus $\varphi: D \to S$.

In dem begleitenden Beispiel (s.o.) ist D_2 ein Schema von D_1, und D_3 ist keines. Die folgende Datenstruktur S_1 ist ebenfalls ein Schema von D_1, und zwar sind D_2 und S_1 isomorph:

S_1: (Person, arbeitet an, Projekt)
(Person, leitet, Projekt)
(Person, arbeitet unter, Person)
(Person, hat als Mitarbeiter, Person).

Dieser Schema-Begriff erlaubt die Charakterisierung dynamisch wachsender, schrumpfender und sich verändernder Mengen von Instance-Strukturen durch ein einziges festes Schema. Die dynamischen Operationen können in naheliegender Weise durch Modifikation des Schema-Homomorphismus beschrieben werden, welche das Bild, also das Schema, konstant lassen. Die dazu notwendigen Modifikationen sind im wesentlichen Erweiterungen und Einschränkungen der Urbildmengen.

Für Homomorphismen von Datenstrukturen läßt sich durch einfache Übertragung bekannter algebraischer Methoden der Homomorphiesatz beweisen: Jeder Homomorphismus $\varphi: D_1 \to D_2$ kann kanonisch zerlegt werden in den kanonischen Homomorphismus $\varkappa: D_1 \to D_1/\pi$ und einen Isomorphismus ι: $D_1/\pi \to D_2$. Das nächste Lemma folgt unmittelbar aus den Definitionen:

Lemma 5.1. Seien D, S Datenstrukturen. S ist genau dann ein Schema von D, wenn es eine schematische Partition π von D gibt, so daß D/π und S isomorph sind.

Bis auf Isomorphie charakterisieren die schematischen Partitionen also vollständig die Menge der möglichen Schemata einer gegebenen Datenstruktur. Der folgende Satz zeigt, daß es in dieser Menge ein speziell ausgezeichnetes Schema gibt.

Satz 5.1. Zu jeder Datenstruktur D gibt es ein bis auf Isomorphie eindeutig bestimmtes „feinstes" Schema S_f von D mit der Eigenschaft, daß jedes Schema S von D auch Schema von S_f ist.

Beweis. Sei $D=(\Omega, A, V, \varrho)$, und sei 0_a die Nullpartition auf A. Nach Satz 4.1 ist $\mathscr{P}(0_a)$ ein Verband, hat also ein Nullelement π_f, und nach Lemma 4.1 ist $\pi_f=(\sigma_\omega(0_a), 0_a, \sigma_v(0_a))$. Sei $S_f:=D/\pi_f$, und sei S irgendein Schema von D mit dem Homomorphismus $\varphi: D\to S$. Sei $D\xrightarrow{\varkappa} D/\pi\xrightarrow{\iota} S$ die kanonische Zerlegung von φ, und sei $\pi=(\pi_\omega, \pi_a, \pi_v)$. Nach Lemma 5.1 ist π schematisch. Daraus und aus Lemma 4.1 folgt: $\pi_\omega \geqq \sigma_\omega(\pi_a)$ und $\pi_v \geqq \sigma_v(\pi_a)$. Nun ist aber trivialerweise $\pi_a \geqq 0_a$, und man sieht leicht, daß dann auch $\sigma_\omega(\pi_a) \geqq \sigma_\omega(0_a)$ sowie $\sigma_v(\pi_a) \geqq \sigma_v(0_a)$ ist. Daraus folgt $\pi \geqq \pi_f$, und es ist $D/\pi=(D/\pi_f)/\tau=S_f/\tau$ für eine Partition τ von S_f. Sei $\varkappa_f: S_f\to S_f/\tau$ der kanonische Homomorphismus. Dann ist $\varphi_f:=\varkappa_f\iota$ ein surjektiver Homomorphismus von S_f auf S, also ist S ein Schema von S_f.

Um zu zeigen, daß S_f bis auf Isomorphie eindeutig bestimmt ist, sei S_f' ein anderes Schema von D mit der Eigenschaft, daß jedes Schema S von D auch Schema von S_f' ist. Dann sind insbesondere S_f und S_f' Schemata voneinander, d.h. es gibt surjektive Homomorphismen $\psi_f: S_f\to S_f'$ und $\psi_f': S_f'\to S_f$. Da ψ_f und ψ_f' endliche Mengen aufeinander abbilden, sind deren Komponenten injektiv, also sind ψ_f und ψ_f' Isomorphismen.

Neben einem „feinsten" Schema für jede Datenstruktur gibt es trivialerweise ein universelles „gröbstes" Schema für alle Datenstrukturen, nämlich

(Objekt, Attribut, Wert).

Literatur

1. Bracchi, G., Fedeli, A., Paolini, P.: A relational data base management system. Proc. ACM Ann. Conf., Boston 1972, p. 1080—1089
2. Childs, D. L.: Feasibility of a set-theoretic data structure: A general structure based on a reconstituted definition of relation. In: Morell, A. J. H. (ed.): Proc. IFIP Congr. 68. Amsterdam: North Holland 1969, p. 420—430
3. CODASYL Data Base Task Group: Report, April 1971
4. CODASYL Systems Commitee: Feature analysis of generalized data base management systems. Technical Report, May 1971
5. Codd, E. F.: A relational model of data for large shared data banks. Comm. ACM **13**, 377—387 (1970)
6. Codd, E. F.: Further normalization of the data base relational model. In [20], p. 33—64
7. Early, J.: Toward an understanding of data structures. Comm. ACM **14**, 617—627 (1971)
8. Early, J.: On the semantics of data structures. In [20], p. 23—32
9. Ehrich, H.-D.: Datenstrukturen und Q-Systeme — eine mathematische Studie. In: Brauer, W. (Hrsg.): Gesellschaft für Informatik, 3. Jahrestagung, Hamburg 1973. Lecture Notes in Computer Science **1**. Berlin-Heidelberg-New York: Springer 1973, p. 363—371
10. Engles, R. W.: A tutorial on data base organization. Annual Review in Automatic Programming **7**, 1—64 (1972)
11. Falkenberg, E. *et al.*: NEOS — Ein Ansatz zur Lösung einiger offener Probleme bei der Konzeption von Datenbanksystemen. In: Deussen, P. (Hrsg.): Gesellschaft für Informatik, 2. Jahrestagung, Karlsruhe 1972. Lecture Notes in Economics and Mathematical Systems **78**. Berlin-Heidelberg-New York: Springer 1973, p. 442—451

12. Feldman, J. A., Rovner, P. D.: An ALGOL-based associative language. Comm. ACM **12**, 439—449 (1969)
13. Hsiao, D., Harary, F.: A formal system for information retrieval from files. Comm. ACM **13**, 67—73 (1970)
14. Neuhold, E. J.: Informationssysteme. In: Deussen, P.(Hrsg.): Gesellschaft für Informatik, 2. Jahrestagung, Karlsruhe 1972. Lecture Notes in Economics and Mathematical Systems **78**. Berlin-Heidelberg-New York: Springer 1973, p. 8—21
15. Notley, M. G.: The Peterlee IS/1 System. UK Scientific Centre, IBM Report UKSC-0018, Peterlee, 1972
16. Rosenberg, A. L.: Data graphs and addressing schemes. J. Computer and System Sciences **5**, 193—238 (1971)
17. Rosenberg, A. L.: Addressable data graphs. J. ACM **19**, 309—430 (1972)
18. Rosenberg, A. L.: Symmetries in data graphs. SIAM J. Computing **1**, 40—65 (1972)
19. Rovner, P. D., Feldman, J. A.: The LEAP language and data structure. In: Morell, A. J. H. (ed.): Proc. IFIP Congr. 68. Amsterdam: North Holland 1969, p. 579—585
20. Rustin, R. (ed.): Data base systems. Courant Computer Science Symposium 6. Englewood Cliffs (N.J.): Prentice-Hall 1972
21. Salton, G.: Automatic information organization and retrieval. New York: McGraw-Hill 1968
22. Schwartz, J. T.: Abstract and concrete problems in the theory of files. In [20], p. 1—21
23. Senko, M. E.: Details of a scientific approach to information systems. In [20], p. 143—174
24. Senko, M. E., Altmann, E. B., Astrahan, M. M., Fehder, P. L.: Data structures and accessing in data base systems. IBM Systems J. **12**, 30—93 (1973)
25. Turski, W. M.: On a model of information retrieval systems based on thesaurus. Information Storage and Retrieval **7**, 89—94 (1971)
26. Turski, W. M.: Data structures and their ordering. IAG J. **3**, 141—150 (1970)
27. Turski, W. M.: A model for data structures and its applications. Part I: Acta Informatica **1**, 26—34 (1971). Part II: Acta Informatica **1**, 282—289 (1972)

H.-D. Ehrich
Abteilung Informatik
Universität Dortmund
D-4600 Dortmund-Hombruch
Postfach 500
Bundesrepublik Deutschland