A Categorical Approach to Database Semantics

Kenneth Baclawski* Dan A. Simovici† William White‡

February 2, 1993

Abstract

We propose a formalization of standard database management systems using topos theory. In this treatment, all constructions take place within an ambient topos which thereby serves as the "universe of discourse." A database schema is defined using objects and morphisms in the ambient topos. A database state for a given schema involves not only the ambient topos but also an internal category within the topos. This approach neatly separates the schema from the state data by placing them in distinct category structures. It is shown that database states can either be regarded syntactically as objects in an external topos or semantically as morphisms in an internal slice category. A number of operations are introduced that correspond to operations used in standard database systems. Extraction selects some of the tables, attributes and domains of a database state. The squeeze operation performs an "elimination of duplicates," which can be combined with extraction to obtain an operation called "projection" in standard relational database systems. A join operation is defined that generalizes the relational join operation and can be used for the cartesian product and selection operations. Finally, "boolean" operations of intersection, union and difference are introduced and related to the other operations.

1 Introduction.

Compared to the field of programming language semantics, which has a sophisticated mathematical foundation, the field of data semantics is unsophisticated and informal. This is especially true of the field of object-oriented databases where the lack of a strong theoretical framework has been frequently noted. However, even for the more established "relational" database systems, there is a gap between relational database systems as they are defined in theory and as they exist in practice. This, in turn, has generated a lack of consistency among the research endeavors in this field. The purpose of this paper is to give a topos-theoretic

*Northeastern University, College of Computer Science, Boston, Massachusetts 02115.

†University of Massachusetts at Boston, Department of Mathematics and Computer Science, Boston, Massachusetts 02125.

‡Intermetrics, Inc., Cambridge, Massachusetts.

treatment of standard (SQL) database systems that is faithful to these systems as they are actually used.

In nearly all data models, a database has two parts: its *schema* or definition and its *state* or current data. Consider the example of a relational database of employees and departments. Its schema might consist of the following:

EMPLOYEES

Attribute	Domain
SSN	CHAR(11)
NAME	CHAR(30)
DEPT	INTEGER
ADDR	STRING

DEPARTMENTS

Attribute	Domain
CODE	INTEGER
NAME	CHAR(20)

The state of the table of employees will look something like the following:

EMPLOYEES

SSN	NAME	DEPT	ADDR
012-00-9091	John Green	1	55 Pleasant St., Brookline, MA 02145
855-48-0712	Ada Byron	2	500 Washington Rd., Newton, MA 02314
670-77-6152	John Green	1	90 Red Rd., Miami, FL 33561
⋮	⋮	⋮	⋮
399-12-1234	Jay Shay	3	4501 Wilshire Blvd., Los Angeles, CA 90901

The state of the table of departments will look something like the following:

DEPARTMENTS

CODE	NAME
1	Product Development
2	Marketing
3	Sales

From this example one can see that a database has a number of distinguishable elements:

1. A schema consists of a number of tables.
2. Each table has a title.
3. Each table has a number of attributes.
4. Each attribute has a domain or set of allowable values.
5. The state of a table has a number of rows (also called "tuples").

6. Each row conforms to the same structure with one value for each attribute. Moreover each value belongs to the domain of the corresponding attribute.

The notion of a schema is fundamental for database management systems. It represents the model or intuitive picture of what one is storing in such a system. Furthermore, the schema itself consists of data that can be organized into tables in much the same way as the data representing the state of the database. The data that make up the schema is sometimes called meta-data to distinguish it from the data in the database.

Of course, one can (and usually does) store the schema as data in its own database (the data dictionary) which is described by a schema of its own (the meta-schema). However, the data in the data dictionary is conceptually on a higher semantic level, and the semantics of the data in the data dictionary is defined by programs in some programming language not by the data itself (whatever that would mean).

In our approach, category theory neatly separates the schema data from the state data by placing them in distinct categories. The schema data reside in an ambient topos (which serves as the "universe of discourse"), while the state data "reside" in an internal category. The conundrum of how one recognizes the data in one level as structures in another is resolved by encoding the data semantics using morphisms in the ambient topos. This is the same technique that is used for defining the internal category itself.

In the next section we review topos theory, giving detailed definitions of internal category and internal functor mostly to illustrate our somewhat eclectic notation. Following this section, the core of the paper is developed in three sections. The concept of a schema is introduced in section 3. We also introduce morphisms of schemas and note that the category of schemas is itself a topos. The main result of this section is the proof that the forgetful functor from internal categories to schemas has an adjoint. Several other forgetful functors are also introduced, and their adjoints are discussed. These other adjoints are used to define a number of forms of "discrete" schema.

The concept of a database state is introduced in section 4, along with the first operation on states: extraction. Because of the adjointness theorem in the previous section, a database state is an internal functor of a certain kind within the ambient topos. Extraction corresponds to the "select" and "from" clauses of an SQL query. It chooses a set of tables, columns and domains from a schema. It can also choose the same object more than once, thereby effectively duplicating it. Database states and extraction form a slice category called the syntactic category of database states to distinguish it from the more interesting category of states defined in the next section. The section concludes with a discussion of states of discrete schemas.

In section 5, an internal (or "semantic") composition is introduced for database states. This composition operation is closely related to the composition of internal natural transformations. With respect to internal composition, the database states are the morphisms of a category called the semantic category of database states. Internal composition is the basic technique for defining semantic notions and operations.

The rest of the paper develops various semantic operations on database states. An operation called the join is introduced in section 6. Categorically, this operation computes

an internal limit in the internal category. Our join operation generalizes the relational equijoin operation which combines two or more tables into a single table. The main result is that joins can be computed whenever the internal category is complete. The join operation allows one to express inter-table constraints such as referential integrity constraints.

In section 7, the concept of squeezing is introduced. This operation corresponds to the "elimination of duplicates" in standard database systems. The main result is a condition on the internal category that ensures that database states can always be squeezed.

An operation called selection is developed in section 8. In standard database systems, this operation selects that part of a database state that satisfies a condition called a "selection predicate." The main result is that the selection operation can be expressed in terms of the join operation. The "boolean" operations of union, intersection and difference are discussed in section 9.

The paper concludes with a discussion of how this framework might be modified to produce an object-oriented data model. In the course of these investigations we found that a standard database schema could be regarded as an incompletely specified category. This led naturally to a forgetful functor and its corresponding adjoint. It came as a surprise that the schema of an object-oriented database appears to be a structure that is intermediate between a standard schema and a category. Put another way, there appears to be a trend toward incorporating more of the structure of a category in the concept of a database schema. Whether this trend will continue is an interesting speculation.

2 Categorical Foundation.

We take as understood the notions of *category*, *functor*, and *natural transformation*. We also assume familiarity with the notions of *completeness* and *cocompleteness*.

To fix notations, If X and A are categories, and $F\ X \to A$ and $G\ A \to X$ are functors, we say F *is left adjoint to* G iff for each pair of objects $x \in X.\texttt{Obj}$ and $a \in A.\texttt{Obj}$ there is a bijection

$$\phi_{x,a}\ X(x, Ga) \to A(Fx, a)$$

. Following MacLane [8], we denote the unit and counit of this adjunction by η and ϵ respectively.

A *topos* forms a universe for engaging in mathematical discourse. In our case it forms the universe for defining database schemas. Technically, a *topos* is a category which is both complete, and in which the subobject functor is representable. However, a naive view of a topos would be a category in which set-theoretic operations are possible. In particular, it is possible to form function spaces using a construction similar to the formation of graphs in ordinary naive set theory. If A and B are objects of a topos, $\mathcal{A}$ then A^B is the object of morphisms from A to B. If $f\ A \to B$ is a morphism of $\mathcal{A}$, then we write $\lceil f \rceil\ 1 \to A^B$ for the (global) element of A^B which represents f.

The universe of discourse provided by a topos is sufficiently rich that one can introduce category theory and even topos theory within the context of a topos. A category defined within a topos is called an *internal category* [7] and is defined as follows.

Definition 1 *An* internal category *C in a topos $\mathcal{A}$ consists of:*

1. *A pair of objects $C.\mathtt{Obj}$ and $C.\mathtt{Arr}$ of $\mathcal{A}$.*

2. *Four morphisms $C.\mathtt{Arr} \xrightarrow{C.\mathtt{source}} C.\mathtt{Obj}$, $C.\mathtt{Arr} \xrightarrow{C.\mathtt{target}} C.\mathtt{Obj}$, $C.\mathtt{Obj} \xrightarrow{C.\mathtt{id}} C.\mathtt{Arr}$ and*

$$C.\mathtt{Arr} \times_{C.\mathtt{Obj}} C.\mathtt{Arr} \xrightarrow{C.\mathtt{comp}} C.\mathtt{Arr},$$

where $C.\mathtt{Arr} \times_{C.\mathtt{Obj}} C.\mathtt{Arr}$ is defined by the following pullback diagram:

$$\begin{array}{ccc} C.\mathtt{Arr} \times_{C.\mathtt{Obj}} C.\mathtt{Arr} & \xrightarrow{\pi_2} & C.\mathtt{Arr} \\ \downarrow{\scriptstyle \pi_1} & & \downarrow{\scriptstyle C.\mathtt{source}} \\ C.\mathtt{Arr} & \xrightarrow{C.\mathtt{target}} & C.\mathtt{Obj} \end{array}$$

The objects and arrows of C satisfy the following constraints:

a. The source and target of an identity arrow is the object on which the identity arrow acts:

$$C.\mathtt{source} \circ C.\mathtt{id} = C.\mathtt{target} \circ C.\mathtt{id} = 1_{C.\mathtt{Obj}}$$

b. The source and target of a composition of arrows is the source of the first arrow and the target of the second arrow, respectively:

$$C.\mathtt{source} \circ C.\mathtt{comp} = C.\mathtt{source} \circ \pi_1 \text{ and } C.\mathtt{target} \circ C.\mathtt{comp} = C.\mathtt{target} \circ \pi_2$$

c. Composition of arrows is associative:

$$C.\mathtt{comp} \circ (1_{C.\mathtt{Arr}} \times_{C.\mathtt{Obj}} C.\mathtt{comp}) = C.\mathtt{comp} \circ (C.\mathtt{comp} \times_{C.\mathtt{Obj}} 1_{C.\mathtt{Arr}})$$

d. The composition of an identity arrow with another arrow is the other arrow:

$$C.\mathtt{comp} \circ (1_{C.\mathtt{Arr}} \times_{C.\mathtt{Obj}} C.\mathtt{id} \circ C.\mathtt{target}) = 1_{C.\mathtt{Arr}}$$

$$C.\mathtt{comp} \circ (C.\mathtt{id} \circ C.\mathtt{source} \times_{C.\mathtt{Obj}} 1_{C.\mathtt{Arr}}) = 1_{C.\mathtt{Arr}}$$

The product morphisms in constraint (d) exist by virtue of constraint (a).

Functors of categories can also be given an internal definition as follows:

Definition 2 *An* internal functor *$f\colon C \to D$ from internal category C to internal category D is a pair of morphisms in the topos $\mathcal{A}$, $f.\mathtt{obj}\colon C.\mathtt{Obj} \to D.\mathtt{Obj}$ and $f.\mathtt{arr}\colon C.\mathtt{Arr} \to D.\mathtt{Arr}$ subject to the following constraints:*

1. *The functor commutes with the source and target morphisms: $f.\mathtt{obj} \circ C.\mathtt{source} = D.\mathtt{source} \circ f.\mathtt{arr}$ and $f.\mathtt{obj} \circ C.\mathtt{target} = D.\mathtt{target} \circ f.\mathtt{arr}$.*

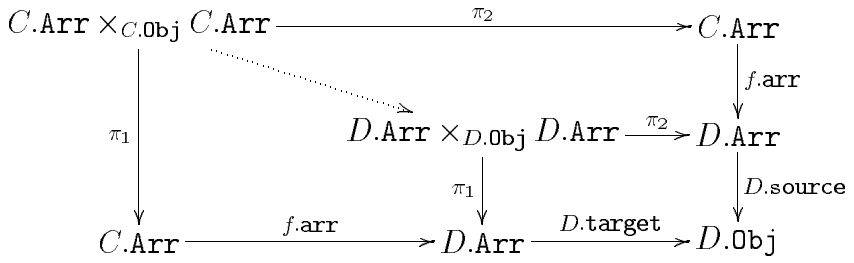


Figure 1: Internal Functors

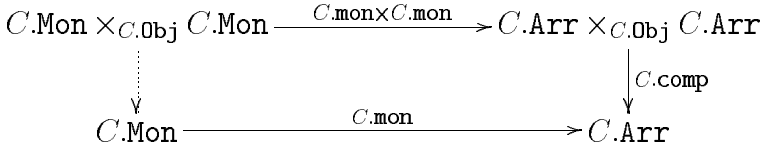
2. *The functor takes identity arrows to identity arrows:* $f.\mathtt{arr} \circ C.\mathtt{id} = D.\mathtt{id} \circ f.\mathtt{obj}$.

3. *The functor preserves composition covariantly:* $f.\mathtt{arr} \circ C.\mathtt{comp} = D.\mathtt{comp} \circ (f.\mathtt{arr} \times f.\mathtt{arr})$, *where the product morphism* $f.\mathtt{arr} \times f.\mathtt{arr}$ *is the morphism* $\langle f.\mathtt{arr} \circ \pi_1, f.\mathtt{arr} \circ \pi_2 \rangle$. *This morphism exists by definition of the pullback* $D.\mathtt{Arr} \times_{D.\mathtt{Obj}} D.\mathtt{Arr}$ *since the diagram in Figure 1 commutes by the constraints in part (1) above and by the definition of the pullback* $C.\mathtt{Arr} \times_{C.\mathtt{Obj}} C.\mathtt{Arr}$.

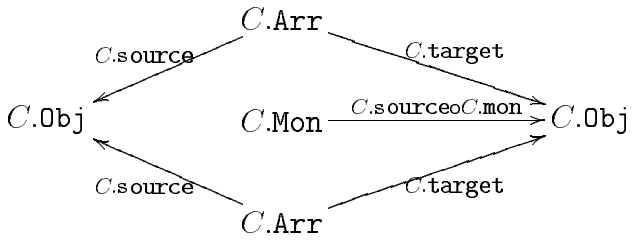
The internal categories and internal functors of the topos $\mathcal{A}$ form an (external) category called the category of internal categories and written $\mathcal{C}at(\mathcal{A})$ or simply $\mathcal{C}at$. One can derive[1] other kinds of internal category from the general definition. Here is an important example.

Definition 3 *An* internal category with monic arrows *is an internal category* C *together with a monomorphism* $C.\mathtt{mon}\colon C.\mathtt{Mon} \to C.\mathtt{Arr}$ *in the ambient topos satisfying the following conditions:*

1. *The morphism* $C.\mathtt{comp} \circ \langle C.\mathtt{mon} \circ \pi_1, C.\mathtt{mon} \circ \pi_2 \rangle$ *factors through* $C.\mathtt{mon}$.



2. *Let* L *be a limit of the following diagram:*



[1]The word "derive" here is used the same way as in the programming languages Ada and C++.

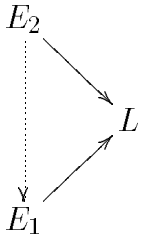
let E_1 be an equalizer of the pair of morphisms

$$L \underset{\pi_3}{\overset{\pi_1}{\rightrightarrows}} C.\mathtt{Arr}$$

and let E_2 be the equalizer of the pair of morphisms

$$L \underset{C.\mathtt{compo}\langle C.\mathtt{mono}\pi_2,\pi_3\rangle}{\overset{C.\mathtt{compo}\langle C.\mathtt{mono}\pi_2,\pi_1\rangle}{\rightrightarrows}} C.\mathtt{Arr}$$

Then there exists a morphism $E_2 \to E_1$ such that this diagram commutes:



Intuitively, an internal category with monic arrows is an internal category with a distinguished set of monic arrows. The first condition states that the composition of two distinguished monic arrows is again a distinguished monic arrow. The second condition states that distinguished monic arrows are monic arrows, i.e., right cancelable. We leave it to the reader to give the definition of an internal category with epic arrows.

One formulation of the axiom of choice is that epimorphisms have right inverses. For internal categories, one postulates this only for the distinguished epic arrows, and a specific right inverse is given for each of these.

Definition 4 *An* internal axiom of choice category *is an internal category with epic arrows together with a morphism $C.\mathtt{choice}\colon C.\mathtt{Epi} \to C.\mathtt{Arr}$ in the ambient topos such that the following diagrams commute:*

$$\begin{array}{ccc} C.\mathtt{Epi} & \xrightarrow{C.\mathtt{epi}} & C.\mathtt{Arr} \\ {\scriptstyle C.\mathtt{choice}}\downarrow & & \downarrow{\scriptstyle C.\mathtt{target}} \\ C.\mathtt{Arr} & \xrightarrow{C.\mathtt{source}} & C.\mathtt{Obj} \end{array} \qquad \begin{array}{ccc} C.\mathtt{Epi} & \xrightarrow{C.\mathtt{epi}} & C.\mathtt{Arr} \\ {\scriptstyle C.\mathtt{choice}}\downarrow & & \downarrow{\scriptstyle C.\mathtt{source}} \\ C.\mathtt{Arr} & \xrightarrow{C.\mathtt{target}} & C.\mathtt{Obj} \end{array}$$

$$\begin{array}{ccc} C.\mathtt{Epi} & \xrightarrow{\langle C.\mathtt{epi},C.\mathtt{choice}\rangle} & C.\mathtt{Arr} \times_{C.\mathtt{Obj}} C.\mathtt{Arr} \\ {\scriptstyle C.\mathtt{target}\circ C.\mathtt{epi}}\downarrow & & \downarrow{\scriptstyle C.\mathtt{comp}} \\ C.\mathtt{Obj} & \xrightarrow{C.\mathtt{id}} & C.\mathtt{Arr} \end{array}$$

3 Schemas

The standard architecture of a database management system has three levels of abstraction (see [6, 10, 11]). This architecture starts with the physical level where data is regarded from the point of view of its representation in memory. It continues with the conceptual level, where the logical and general view of data in the database is represented in a conceptual schema. It concludes with the external level, which includes all individual views of the users of the database. In this section we focus on a categorical formulation on the notion of schema. Our definition is meant to capture the main organizing data structures (referred below as the *tops*), the properties of various objects of the database (called *attributes*) and the sets which allow us to express these properties (called *domains*).

Definition 5 *A* schema *in the ambient topos $\mathcal{A}$ is a quintuple $S = (T, A, F, \tau, \phi)$ for which T,A and F are objects in $\mathcal{A}$, and for which $\tau\colon A \to T$ and $\phi\colon A \to F$ are morphisms.*

The object T of a schema is called the *set of tops*. Intuitively, the elements of T are the *names* or *labels* of tables. The elements are *not* the tables themselves. As will be seen shortly, a table may be regarded as a particular *state* of a top. For a schema S, the set of tops will be denoted $S.\texttt{Top}$.

The object A of a schema is called the *set of attributes*. The morphism $\tau\colon A \to T$ may be regarded as the function that maps an attribute to its corresponding top. Again, one should not confuse an attribute with a *column* which is a state of an attribute. Note that each attribute belongs to just one top. In some treatments of the relational model, it is possible for an attribute to belong to several relations. From this point of view, an element of A is a *fully qualified attribute*, where the qualification specifies the relation name of the attribute. For a schema S, the set of attributes will be denoted $S.\texttt{Attributes}$, and the morphism $\tau\colon S.\texttt{Attributes} \to S.\texttt{Top}$ will be written $S.\texttt{top}$.

The object F of a schema is called the *set of feet*. Intuitively, the elements of F are the *names* or *labels* of domains (or types). The elements of F are *not* the domains themselves. As in the case of tops and attributes, a domain may be regarded as a particular *state* of a foot. The morphism $\phi\colon A \to F$ may be regarded as the function that maps an attribute to the name of its type. For a schema S, the set of feet will be denoted $S.\texttt{Foot}$, and the morphism $\phi\colon S.\texttt{Attributes} \to S.\texttt{Foot}$ will be denoted $S.\texttt{foot}$.

Example 6 *Consider the database of employees and departments in section 1. Call the schema S. The schema has two top elements,*

$$S.\texttt{Top} = \{EMPLOYEES, DEPARTMENTS\},$$

but it has six attributes,

$$S.\texttt{Attributes} = \{E.SSN, E.NAME, E.DEPT, E.ADDR, D.CODE, D.NAME\},$$

where E is an abbreviation for EMPLOYEES and D abbreviates DEPARTMENTS. The schema also has five feet,

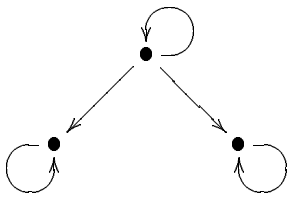
$$S.\texttt{Foot} = \{INTEGER, CHAR(11), CHAR(20), CHAR(30), STRING\}.^2$$

The morphism $S.\texttt{Attributes} \xrightarrow{S.\texttt{top}} S.\texttt{Top}$ *maps E.SSN to EMPLOYEES, ..., D.NAME to DEPARTMENTS. Similarly,* $S.\texttt{Attributes} \xrightarrow{S.\texttt{foot}} S.\texttt{Foot}$ *maps E.SSN to CHAR(11), and so on.*

So far the definition of a schema treats domains and tables symmetrically – both are part of the state. In fact, the set of domains is usually considered to be fixed and the state of the database is entirely determined by the states of the tables. This feature of a schema is discussed later (cf. definition 24 of section 5) when the concept of semantic isomorphism of states is introduced.

The concept of a schema defined above allows the possibility of a top element that has no attributes at all. Although such a top element is not especially useful, it isn't unreasonable to allow it.

A schema is a cone in the ambient topos or, equivalently, an object in the category of functors into the ambient topos from the category having three objects and five morphisms as in this diagram:



A *morphism of schemas* is a morphism (natural transformation) in the functor category mentioned above. In other words, a morphism of schemas $m\colon S \to T$ is a triple of morphisms (t, a, f) in the ambient topos such that the following diagram commutes:

$$\begin{array}{ccc} S.\texttt{Top} & \xrightarrow{t} & T.\texttt{Top} \\ {\scriptstyle S.\texttt{top}}\uparrow & & \uparrow{\scriptstyle T.\texttt{top}} \\ S.\texttt{Attributes} & \xrightarrow{a} & T.\texttt{Attributes} \\ {\scriptstyle S.\texttt{foot}}\downarrow & & \downarrow{\scriptstyle T.\texttt{foot}} \\ S.\texttt{Foot} & \xrightarrow{f} & T.\texttt{Foot} \end{array} \qquad (1)$$

The schemas and morphisms of schemas of the topos $\mathcal{A}$ form a category called the category of schemas, written $\mathcal{S}ch(\mathcal{A})$ or simply $\mathcal{S}ch$. This category is actually a topos, although we will not use this fact. See [7, 2].

[2]Actually, the notation in section 1 is not a complete specification of a schema in the sense used in this section. It does not specify that a domain such as CHAR(5) is in the schema. Such a domain is potentially usable, but was not explicitly used.

It is convenient to refer to a morphism of schemas by a single name from which the three components are extracted. If $m = (t, a, f)$ is a morphism of schemas, then its three components will be denoted by $m.\texttt{tables}$, $m.\texttt{columns}$ and $m.\texttt{domains}$, respectively. The significance of this notation will become clear after the concept of a database state has been introduced in section 4.

Note that there are some similarities between a schema and an internal category, where the morphism $S.\texttt{top}$ is analogous to the C.`source` morphism of an internal category and the morphism $S.\texttt{foot}$ is analogous to the C.`target` morphism of an internal category. The main difference is that the objects $S.\texttt{Top}$ and $S.\texttt{Foot}$ need not coincide, as they must in an internal category. As a result, there are no concepts of composition and identity internal to a schema. Nevertheless, it is helpful to regard a schema as an incomplete specification for a category. In particular, there is a forgetful functor $\mathcal{U}$ that regards an internal category as a schema.

Definition 7 *There is a "forgetful functor" $\mathcal{U}$:$\mathcal{C}at \to \mathcal{S}ch$ given on an internal category C by:*

(1) $\mathcal{U}(C).\texttt{Top} = \mathcal{U}(C).\texttt{Foot} = C.\texttt{Obj}$,

(2) $\mathcal{U}(C).\texttt{Attributes} = C.\texttt{Arr}$,

(3) $\mathcal{U}(C).\texttt{top} = C.\texttt{source}$,

(4) $\mathcal{U}(C).\texttt{foot} = C.\texttt{target}$.

and on an internal functor f by:

(1) $\mathcal{U}(f).\texttt{tables} = f.\texttt{obj}$,

(2) $\mathcal{U}(f).\texttt{columns} = f.\texttt{arr}$,

(3) $\mathcal{U}(f).\texttt{domains} = f.\texttt{obj}$.

The forgetful functor $\mathcal{U}$ has an adjoint that is important for defining the concept of a database state.

Theorem 8 *The functor $\mathcal{U}$:$\mathcal{C}at \to \mathcal{S}ch$ has a left adjoint $\mathcal{F}$:$\mathcal{S}ch \to \mathcal{C}at$.*

Before proving the theorem, we begin with some useful auxiliary results.

Lemma 9 *Let A and B be two objects in the topos $\mathcal{A}$. The following is a pullback square:*

$$\begin{array}{ccc} 0 & \longrightarrow & B \\ \downarrow & & \downarrow{\scriptstyle \iota_2} \\ A & \xrightarrow{\iota_1} & A + B \end{array}$$

Proof. It is trivial to see that

$$\begin{array}{ccc} 0 & \longrightarrow & 0 \\ \downarrow & & \downarrow \\ A & \longrightarrow & 1 \end{array} \qquad (2)$$

is a pullback square. Furthermore, since $0 \longrightarrow B$ is monic, the definition of the subobject classifier implies the existence of a unique morphism $false_B$ such that

$$\begin{array}{ccc} 0 & \longrightarrow & B \\ \downarrow & & \downarrow{\scriptstyle false_B} \\ 1 & \xrightarrow{true} & \Omega \end{array} \tag{3}$$

Combining diagrams 2 and 3 and using the Pullback Lemma [5, page 67] gives the pullback square:

$$\begin{array}{ccc} 0 & \longrightarrow & B \\ \downarrow & & \downarrow{\scriptstyle false_B} \\ A & \xrightarrow{true_A} & \Omega \end{array} \tag{4}$$

By definition of the coproduct, there is a unique coproduct morphism

$$[true_A, false_B]\colon A + B \to \Omega.$$

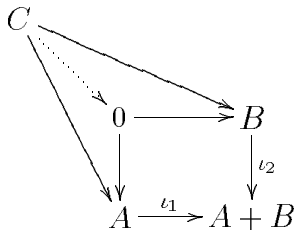
Suppose that the following is a commutative square:

$$\begin{array}{ccc} C & \longrightarrow & B \\ \downarrow & & \downarrow{\scriptstyle \iota_2} \\ A & \xrightarrow{\iota_1} & A + B \end{array} \tag{5}$$

By definition of the coproduct morphism, the commutativity of diagram 5 implies the following square:

$$\begin{array}{ccc} C & \longrightarrow & B \\ \downarrow & & \downarrow{\scriptstyle false_B} \\ A & \xrightarrow{true_A} & \Omega \end{array}$$

also commutes. Since diagram 4 is a pullback, there is a unique morphism $C \to 0$ such that



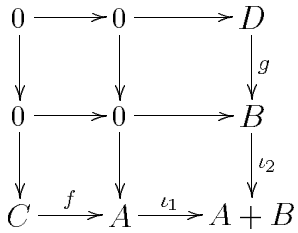
commutes. The lemma then follows. ∎

Corollary 10 *Let $f\colon C \to A$ and $g\colon D \to B$ be morphisms. Then*

$$\begin{array}{ccc} 0 & \longrightarrow & D \\ \downarrow & & \downarrow{\scriptstyle \iota_2 \circ g} \\ C & \xrightarrow{\iota_1 \circ f} & A + B \end{array}$$

is a pullback square.

Proof. In the following diagram:



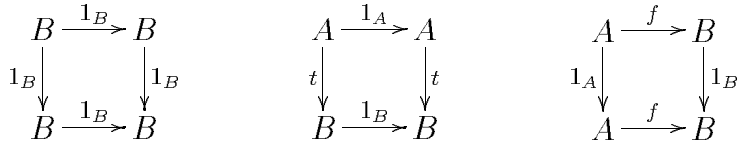
the lower right square is the pullback square in Lemma 9, while the other squares are trivially pullback squares. The Corollary follows from the Pullback Lemma. ∎

Proposition 11 *Let $t: A \to T$ and $f: A \to F$ be two morphisms. Then the diagram:*

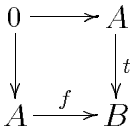
$$\begin{array}{ccc} A+T+F+A & \xrightarrow{[\iota_1,\iota_2,\iota_3,\iota_3 \circ f]} & A+T+F \\ {\scriptstyle [\iota_1 \circ t,\iota_1,\iota_2,\iota_3]}\downarrow & & \downarrow{\scriptstyle [\iota_1 \circ t,\iota_1,\iota_2]} \\ T+F+A & \xrightarrow{[\iota_1,\iota_2,\iota_2 \circ f]} & T+F \end{array}$$

is a pullback square.

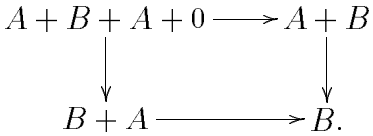
Proof. Let $B = T + F$. Clearly the following are all pullback squares, where f is short for $\iota_2 \circ f$ and t is short for $\iota_1 \circ t$:



By Corollary 10, the following diagram is also a pullback square:



It is a well-known consequence of the Fundamental Theorem of Topoi that coproducts preserve pullbacks (see [3]). The four pullback squares above may therefore be combined into a single pullback square:



Since $A + B + A + 0 \cong A + B + A$, the Proposition follows. ∎

Proof of Theorem 8. The adjoint functor $\mathcal{F}: \mathcal{S}ch \to \mathcal{C}at$ is defined on a schema S as follows:

1. $\mathcal{F}(S).\texttt{Obj}$ is the coproduct $S.\texttt{Top} + S.\texttt{Foot}$.

2. $\mathcal{F}(S).\texttt{Arr}$ is the coproduct $S.\texttt{Attributes} + S.\texttt{Top} + S.\texttt{Foot}$.

3. $\mathcal{F}(S).\texttt{source}$ is the coproduct morphism

$$[\iota_1 \circ S.\texttt{top}, \iota_1, \iota_2]\colon S.\texttt{Attributes} + S.\texttt{Top} + S.\texttt{Foot} \to S.\texttt{Top} + S.\texttt{Foot}$$

4. $\mathcal{F}(S).\texttt{target}$ is the coproduct morphism

$$[\iota_2 \circ S.\texttt{foot}, \iota_1, \iota_2]\colon S.\texttt{Attributes} + S.\texttt{Top} + S.\texttt{Foot} \to S.\texttt{Top} + S.\texttt{Foot}$$

5. $\mathcal{F}(S).\texttt{id}$ is the coproduct morphism

$$[\iota_2, \iota_3]\colon S.\texttt{Top} + S.\texttt{Foot} \to S.\texttt{Attributes} + S.\texttt{Top} + S.\texttt{Foot}$$

6. By Proposition 11,

$$\mathcal{F}(S).\texttt{Arr} \times_{\mathcal{F}(S).\texttt{Obj}} \mathcal{F}(S).\texttt{Arr}$$

   is isomorphic to $S.\texttt{Attributes} + S.\texttt{Top} + S.\texttt{Foot} + S.\texttt{Attributes}$. $\mathcal{F}(S).\texttt{comp}$ is then defined by $[\iota_1, \iota_2, \iota_3, \iota_1]$.

It is straightforward to verify that $\mathcal{F}(S)$ is an internal category. To complete the definition of $\mathcal{F}$ as a functor, we need to define $\mathcal{F}(f)$ for a morphism of schemas $f\colon S \to T$. $\mathcal{F}(f)$ is the internal functor from $\mathcal{F}(S)$ to $\mathcal{F}(T)$ given by:

1. $\mathcal{F}(f).\texttt{Obj}\colon \mathcal{F}(S).\texttt{Obj} \to \mathcal{F}(T).\texttt{Obj}$ is the morphism

$$[\iota_1 \circ f.\texttt{tables}, \iota_2 \circ f.\texttt{domains}]\colon S.\texttt{Top} + S.\texttt{Foot} \to T.\texttt{Top} + T.\texttt{Foot}$$

2. $\mathcal{F}(f).\texttt{arr}\colon \mathcal{F}(S).\texttt{Arr} \to \mathcal{F}(T).\texttt{Arr}$ is the morphism

$$[\iota_1 \circ f.\texttt{columns}, \iota_2 \circ f.\texttt{tables}, \iota_3 \circ f.\texttt{domains}]$$

Again, it is straightforward to verify that $\mathcal{F}(f)$ is an internal functor. The next step is to define a morphism of schemas $f_S\colon S \to \mathcal{U}(\mathcal{F}(S))$ for every schema S. This morphism is given by:

1. $f_S.\texttt{tables}$ is the injection $\iota_1\colon S.\texttt{Top} \to S.\texttt{Top} + S.\texttt{Foot}$,

2. $f_S.\texttt{columns}$ is the injection $\iota_1\colon S.\texttt{Attributes} \to S.\texttt{Attributes} + S.\texttt{Top} + S.\texttt{Foot}$,

3. $f_S.\texttt{domains}$ is the injection $\iota_2\colon S.\texttt{Foot} \to S.\texttt{Top} + S.\texttt{Foot}$,

Clearly, f_S is a morphism of schemas. Let C be an internal category, and $g\colon S \to \mathcal{U}(C)$ be a morphism of schemas. Define a functor $\hat{g}\colon \mathcal{F}(S) \to C$ as follows:

1. $\hat{g}.\mathtt{obj} = [g.\mathtt{tables}, g.\mathtt{domains}]\colon S.\mathtt{Top} + S.\mathtt{Foot} \to C.\mathtt{Obj}$.

2. $\hat{g}.\mathtt{arr}\colon S.\mathtt{Attributes} + S.\mathtt{Top} + S.\mathtt{Foot} \to C.\mathtt{Arr}$ is the morphism $[g.\mathtt{columns}, C.\mathtt{id} \circ g.\mathtt{tables}, C.\mathtt{id} \circ g.\mathtt{domains}]$.

It is straightforward, but tedious, to check that $\hat{g}$ is an internal functor and that $g = \mathcal{U}(\hat{g}) \circ f_S$. All that remains is to check that $\hat{g}$ is unique subject to these properties. Accordingly, let $h\colon \mathcal{F}(S) \to C$ be an internal functor such that $g = \mathcal{U}(h) \circ f_S$. This implies that $g.\mathtt{tables} = (\mathcal{U}(h) \circ f_S).\mathtt{tables} = h.\mathtt{obj} \circ f_S.\mathtt{tables} = h.\mathtt{obj} \circ \iota_1$. Similarly, $g.\mathtt{domains} = h.\mathtt{obj} \circ \iota_2$. By the uniqueness of the coproduct morphism, $h.\mathtt{obj} = [g.\mathtt{tables}, g.\mathtt{domains}] = \hat{g}.\mathtt{obj}$.

In a similar manner, we compute $g.\mathtt{columns} = h.\mathtt{arr} \circ \iota_1$. Now h is an internal functor so this diagram commutes:

$$\begin{array}{ccc} S.\mathtt{Top} + S.\mathtt{Foot} & \xrightarrow{\mathcal{F}(S).\mathtt{id}=[\iota_2,\iota_3]} & S.\mathtt{Attributes} + S.\mathtt{Top} + S.\mathtt{Foot} \\ {\scriptstyle \hat{g}.\mathtt{obj}=h.\mathtt{obj}}\downarrow & & \downarrow{\scriptstyle h.\mathtt{arr}} \\ C.\mathtt{Obj} & \xrightarrow{C.\mathtt{id}} & C.\mathtt{Arr} \end{array}$$

Therefore, $h.\mathtt{arr} \circ \iota_2 = C.\mathtt{id} \circ g.\mathtt{tables}$ and $h.\mathtt{arr} \circ \iota_3 = C.\mathtt{id} \circ g.\mathtt{domains}$. Again, by the uniqueness of the coproduct morphism, $h.\mathtt{arr} = [g.\mathtt{columns}, C.\mathtt{id} \circ g.\mathtt{tables}, C.\mathtt{id} \circ g.\mathtt{domains}] = \hat{g}.\mathtt{arr}$. ∎

There are several other forgetful functors that have adjoints. An internal category C can be regarded as an object of $\mathcal{A}$ by forgetting all of the structure of C except the object $C.\mathtt{Arr}$. Write $\mathcal{V}$ for this forgetful functor. Similarly, a schema S can be regarded as an object of $\mathcal{A}$ by forgetting all but $S.\mathtt{Attributes}$. Write $\mathcal{W}$ for this forgetful functor. Note that $\mathcal{W}(\mathcal{U}(C)) = \mathcal{V}(C)$. The left adjoints of $\mathcal{V}$ and $\mathcal{W}$ will both be denoted $\mathcal{D}$. For an object A of $\mathcal{A}$, $\mathcal{D}(A)$ is called the *discrete internal category* (or *discrete schema*, depending on the context) of the object A. In a discrete internal category, all the arrows are identity arrows. In a discrete schema, the tops, attributes and feet all coincide.

Another useful forgetful functor and left adjoint is the one that regards a schema as a morphism of $\mathcal{A}$. The forgetful functor is given by $\mathcal{U}_t(S) = S.\mathtt{top}$ for a schema S, and is defined on morphisms of schemas in the obvious way. $\mathcal{U}_t$ is a functor from $\mathcal{S}ch$ to the morphism category $\mathcal{A}^{\to}$ of $\mathcal{A}$. (There is also a forgetful functor $\mathcal{U}_f(S) = S.\mathtt{foot}$, but we will not have a use for this one.) The left adjoint of $\mathcal{U}_t$ is $\mathcal{F}_t$, which is defined on a morphism $f\colon A \to B$ by:

1. $\mathcal{F}_t(f).\mathtt{Top} = B$,
2. $\mathcal{F}_t(f).\mathtt{Attributes} = A$,
3. $\mathcal{F}_t(f).\mathtt{Foot} = A$,
4. $\mathcal{F}_t(f).\mathtt{top} = f$,
5. $\mathcal{F}_t(f).\mathtt{foot} = 1_A$.

4 Database States and Extraction

The notion of database state (see [10]) is introduced in order to represent the content of the database at a given moment in time. Also, in this section we consider the extraction operation on database states. This operation is a generalization of both aliasing and the projection operation from relational databases.

Definition 12 *A* (database) state *of a schema S in an internal category C is a morphism of schemas from S to $\mathcal{U}(C)$.*

A state s of a schema S consists of three morphisms. The first, $S.\texttt{Top} \xrightarrow{s.\texttt{tables}} C.\texttt{Obj}$, instantiates each top as a table. The second, $S.\texttt{Attributes} \xrightarrow{s.\texttt{columns}} C.\texttt{Arr}$, instantiates each attribute as a column, i.e., as a function from a table to a domain. The third, $S.\texttt{Foot} \xrightarrow{s.\texttt{domains}} C.\texttt{Obj}$, instantiates each foot as a domain. The commutativity of diagram 1 of section 3 ensures that the column of an attribute maps the table of the top of the attribute to the domain of the foot of the attribute.

Example 13 *Consider again the table of employees in the introduction. The schema S of this table was defined in Example 6. A state s of this schema consists of three morphisms:*

1. A morphism

   $$s.\texttt{tables}\colon \{EMPLOYEES, DEPARTMENTS\} \to C.\texttt{Obj}.$$

   The image of $EMPLOYEES$, for example, is the set of rows of the EMPLOYEES table. To write this set down, one would need some kind of identifier for each row. This identifier is not usually regarded as part of the database state. Later we will discuss the concept of isomorphic states which will deal with this issue. For the purpose of this example, suppose that $s.\texttt{tables}(EMPLOYEES)$ is the set $\{1, 2, 3, \ldots, n\}$, where n is the number of rows in the table.

2. A morphism

   $$\{\text{INTEGER,CHAR(11),CHAR(20),CHAR(30),STRING}\} \xrightarrow{s.\texttt{domains}} \text{C.}\texttt{Obj}$$

   This morphism defines these five domains. For example, $s.\texttt{domains}$ maps the label $CHAR(11)$ to the set of 11-character strings in $C.\texttt{Obj}$.

3. A morphism

   $$\{E.SSN, E.NAME, E.DEPT, E.ADDR, D.CODE, D.NAME\} \xrightarrow{s.\texttt{columns}} C.\texttt{Arr}$$

   The image of $E.NAME$, for example, is a function that maps the elements of the set $s.\texttt{tables}(EMPLOYEES)$ to the set of 11-character strings. In the table shown in the introduction, this is the function $\{1 \mapsto \texttt{"John Green"}, 2 \mapsto \texttt{"Ada Byron"}, 3 \mapsto \texttt{"John Green"}, \ldots, n \mapsto \texttt{"Jay Shay"}\}$.

Because the forgetful functor $\mathcal{U}$ has a left adjoint $\mathcal{F}$, one could equally well define a database state as an (internal) functor $\mathcal{F}(S) \to C$. This is the origin of the title of the paper.

The first operation one can perform on database states is called *extraction*. It selects some of the tables, columns and domains of a database state.

Definition 14 *Let S and S' be two schemas, and let $m\colon S' \to S$ be a morphism of schemas. Let $s\colon S \to \mathcal{U}(C)$ be a state of S. The* extraction *of s via m is the composition of morphisms $m^*(s) = s \circ m\colon S' \to \mathcal{U}(C)$.*

The extraction operation is similar to the relational projection operation, but there are a number of differences. For example, it deals with an entire schema not just a single relation. Furthermore, one can select tables, columns and domains more than once, i.e., one can "alias" any of these concepts. Finally, the extraction does not "eliminate duplicates."

For a fixed internal category C, the database states and extractions form a category. To be more precise, the objects of this category are the database states $s\colon S \to \mathcal{U}(C)$. The morphisms are pairs of the form (m, s), where m is a morphism $m\colon T \to S$ of schemas and s is a state $s\colon S \to \mathcal{U}(C)$ of S in C. The resulting category is better known as the *slice category* $\mathcal{S}ch/\mathcal{U}(C)$ of $\mathcal{S}ch$ over $\mathcal{U}(C)$. See, for example, [8]. We call this the *syntactic category of database states* in C. As we noted already, $\mathcal{S}ch$ is a topos. Therefore the syntactic category is also a topos. The term "syntactic" was chosen to contrast the extraction operation, which is external, with the "semantic" or internal operations to be defined later.

Example 15 *Consider the discrete schema $\mathcal{D}(A)$ for an object A of $\mathcal{A}$. A state of this schema is completely determined by the morphism $A \to C.\mathtt{Arr}$. In other words, a state of a discrete schema is just a collection of unrelated arrows. This is an important special case, and such a state will be called a* discrete state.

Definition 16 *Let A be a discrete schema. A state i of A is said to be a* discrete identity state *in C if the morphism $i.\mathtt{columns}\colon A \to C.\mathtt{Arr}$ factors through the morphism $C.\mathtt{id}$ of the internal category C.*

A discrete identity state of A is determined by a morphism $A \to C.\mathtt{Obj}$. Intuitively, a discrete identity state is a collection of identity arrows in the internal category or, equivalently, a collection of internal objects of the internal category. In contrast to the extraction operation, this is an internal, semantic concept.

More generally, for a fixed object A of $\mathcal{A}$, the discrete states form a category:

Definition 17 *Let A be an object of $\mathcal{A}$. The database states of $\mathcal{D}(A)$ in an internal category C form an internal category denoted C^A, where $C^A.\mathtt{Obj}$ is the exponential object $= C.\mathtt{Obj}^A$ of $\mathcal{A}$, $C^A.\mathtt{Arr} = C.\mathtt{Arr}^A$, $C^A.\mathtt{source}$ is the exponential morphism $C.\mathtt{source}^A$, and so on.*

It is easy to check that C^A is an internal category. The discrete states of $\mathcal{D}(A)$ in C are in bijective correspondence with the morphisms from 1 to $C^A.\mathtt{Arr}$. The state $s\colon \mathcal{D}(A) \to \mathcal{U}(C)$

corresponds to the morphism $\lceil s.\texttt{columns}\rceil : 1 \to C.\texttt{Arr}^A$. We will often abuse notation and refer to the morphism $s.\texttt{columns}$ as being the state s. The discrete identity states are bijective with morphisms from 1 to $C^A.\texttt{Obj}$. A discrete state s is a discrete identity state if $C^A.\texttt{id} \circ \lceil s.\texttt{tables}\rceil = \lceil s.\texttt{columns}\rceil$. A discrete state s is a *discrete isomorphism state* if there is another discrete state t such that $C^A.\texttt{comp} \circ \langle \lceil s\rceil, \lceil t\rceil\rangle$ and $C^A.\texttt{comp} \circ \langle \lceil t\rceil, \lceil s\rceil\rangle$ are both well-defined discrete identity states. Other internal concepts for discrete states can easily be defined by this technique.

5 Internal Composition of States.

We now introduce a (semantic) composition operation for database states which generalizes the internal composition operation for discrete states. This form of composition should be contrasted with the syntactic notion of composition used in the definition of extraction. Internal composition allows one to define isomorphism of states as well as relational algebra operations such as join, selection and projection. We first introduce a schema composition operation for schemas and morphisms of schemas.

Definition 18 *A schema Q is said to be* compatible *with a schema R if $Q.\texttt{Foot} = R.\texttt{Top}$. When Q is compatible with R, the* schema composition schema *is the schema S, denoted $R \otimes Q$ given as follows:*

1. *S is the schema such that $S.\texttt{Top} = Q.\texttt{Top}$, $S.\texttt{Foot} = R.\texttt{Foot}$ and $S.\texttt{Attributes}$ is given by the following pullback in the ambient topos:*

$$\begin{array}{ccc} S.\texttt{Attributes} & \xrightarrow{\pi_1} & Q.\texttt{Attributes} \\ \downarrow{\scriptstyle \pi_2} & & \downarrow{\scriptstyle Q.\texttt{foot}} \\ R.\texttt{Attributes} & \xrightarrow{R.\texttt{top}} & Q.\texttt{Foot} \end{array} \tag{6}$$

2. *The morphism $S.\texttt{top}$ is given by $R.\texttt{top} \circ \pi_2$, and the morphism $S.\texttt{foot}$ is given by $Q.\texttt{foot} \circ \pi_1$.*

The full diagram defining the schema composition of Q with R is Figure 2. Our notion of schema composition of schemas has appeared in other contexts. For example, the composition of partial maps is a special case. See, for example, [1], pages 34–35.

A number of special cases of schema composition are worth noting. For a schema S, the schemas $\mathcal{D}(S.\texttt{Foot}) \otimes S$ and $S \otimes \mathcal{D}(S.\texttt{Top})$ are both naturally isomorphic to S, and for simplicity we identify both of them with S.

Example 19 *The schema of an internal category $\mathcal{U}(C)$ is always compatible with itself. In this case, $(\mathcal{U}(C) \otimes \mathcal{U}(C)).\texttt{Attributes} = C.\texttt{Arr} \times_{C.\texttt{Obj}} C.\texttt{Arr}$ is the "set of composable arrows of C." The composition of these composable arrows defines a database state. We overload our notation somewhat and denote this state by $C.\texttt{comp}: \mathcal{U}(C) \otimes \mathcal{U}(C) \to \mathcal{U}(C)$. This state is*

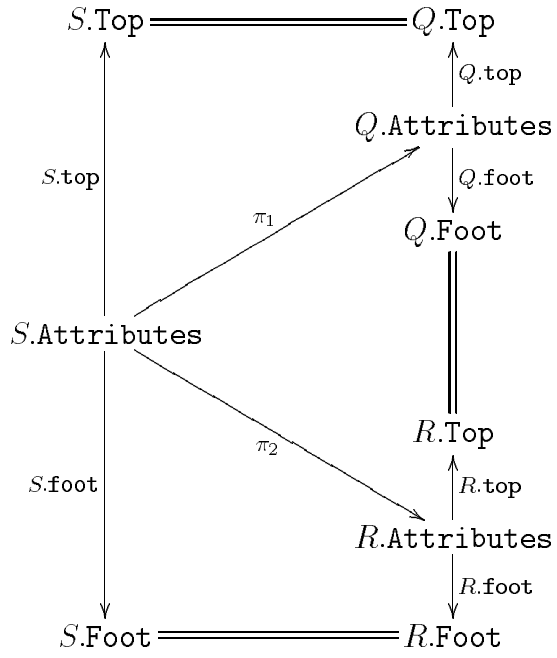


Figure 2: Schema Composition

the identity on tops and feet so the composition on arrows is the only nontrivial effect as a morphism of schemas. This justifies the overloading of $C.\mathtt{comp}$ *for this morphism. It is easy to check using the definition of an internal category that* $C.\mathtt{comp}$ *is a morphism.*

Because a morphism of schemas has three components, there are many ways to factor it through intermediate schemas. The following factorization of a morphism $f: S \to T$ will be important in the definition of the join in Section 6:

$$S \longrightarrow S \otimes \mathcal{F}_t(f.\mathtt{tables}) \longrightarrow T$$

$$\begin{array}{ccccc}
S.\mathtt{Top} & \xrightarrow{f.\mathtt{tables}} & T.\mathtt{Top} & = & T.\mathtt{Top} \\
\uparrow{\scriptstyle S.\mathtt{top}} & & \uparrow{\scriptstyle f.\mathtt{tables} \circ S.\mathtt{top}} & & \uparrow{\scriptstyle T.\mathtt{top}} \\
S.\mathtt{Attributes} & = & S.\mathtt{Attributes} & \xrightarrow{f.\mathtt{columns}} & T.\mathtt{Attributes} \\
\downarrow{\scriptstyle S.\mathtt{foot}} & & \downarrow{\scriptstyle S.\mathtt{foot}} & & \downarrow{\scriptstyle T.\mathtt{foot}} \\
S.\mathtt{Foot} & = & S.\mathtt{Foot} & \xrightarrow{f.\mathtt{domains}} & T.\mathtt{Foot}.
\end{array} \tag{7}$$

We now define the schema composition operation on morphisms of schemas.

Definition 20 *Let* $m: P \to S$ *and* $n: Q \to T$ *be two morphisms of schemas. The morphisms* m *and* n *are said to be* compatible *when* $m.\mathtt{domains} = n.\mathtt{tables}$. *When* m *and* n *are compatible, their* schema composition *is the morphism* $n \otimes m: Q \otimes P \to T \otimes S$ *given by:*

1. $(n \otimes m).\texttt{tables} = m.\texttt{tables}$

2. $(n \otimes m).\texttt{domains} = m.\texttt{domains}$

3. $(n \otimes m).\texttt{columns} = \langle m.\texttt{columns} \circ \pi_1, n.\texttt{columns} \circ \pi_2 \rangle$

It is easy to verify that $n \otimes m$ is a morphism of schemas.

An important property of the $\otimes$ operator is that it is natural with respect to composition in the ambient topos. This is a fundamental fact that is the basis for relating the external (syntactic) composition of schema morphisms with internal (semantic) operations of database states.

Proposition 21 *Let $P \xrightarrow{p} Q \xrightarrow{q} R$ and $S \xrightarrow{s} T \xrightarrow{t} U$ be schema morphisms such that p and s as well as q and t are compatible. Then $(t \otimes q) \circ (s \otimes p) = (t \circ s) \otimes (q \circ p)$.*

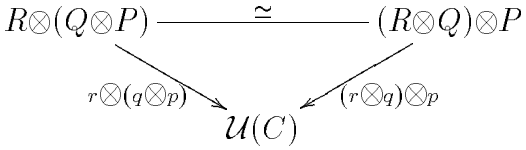
Proof. Follows easily from the corresponding property for pullback. ∎

The schema composition of morphisms can, of course, be applied to database states because a state is a certain kind of morphism of schemas. However, the result of such an operation on states is no longer a state. Accordingly, we need to modify the construction to obtain an operation that again produces a state. We call this operation the internal composition of database states.

Definition 22 *Let $q: Q \to \mathcal{U}(C)$ and $r: R \to \mathcal{U}(C)$ be two states in the internal category C that are compatible as morphisms of schemas. The* internal composition *of q with r, is given by $r \circledast q = C.\texttt{comp} \circ (r \otimes q)$, where $C.\texttt{comp}$ was defined in Example 19.*

Combining Figure 2 with the commutative diagrams for the states q and r gives a commutative diagram in the ambient topos shown in Figure 3 that defines $r \circledast q$. An important feature of internal composition of states is that they are associative up to isomorphism.

Theorem 23 *Let p, q, r be states of the schemas P, Q, R, respectively, in C such that p is compatible with q and q is compatible with r. Then there is an isomorphism of schemas such that the following diagram commutes:*



Proof. The morphisms $(r \circledast (q \circledast p)).\texttt{tables}$ $((r \circledast q) \circledast p).\texttt{tables}$ are both equal to $p.\texttt{tables}$ and so coincide. The .domains components also coincide, so the only problematic morphisms are the .columns components. Now $(R \otimes (Q \otimes P)).\texttt{Attributes}$ and $((P \otimes Q) \otimes R).\texttt{Attributes}$ are given by pullbacks which are known to be naturally isomorphic. If we abbreviate

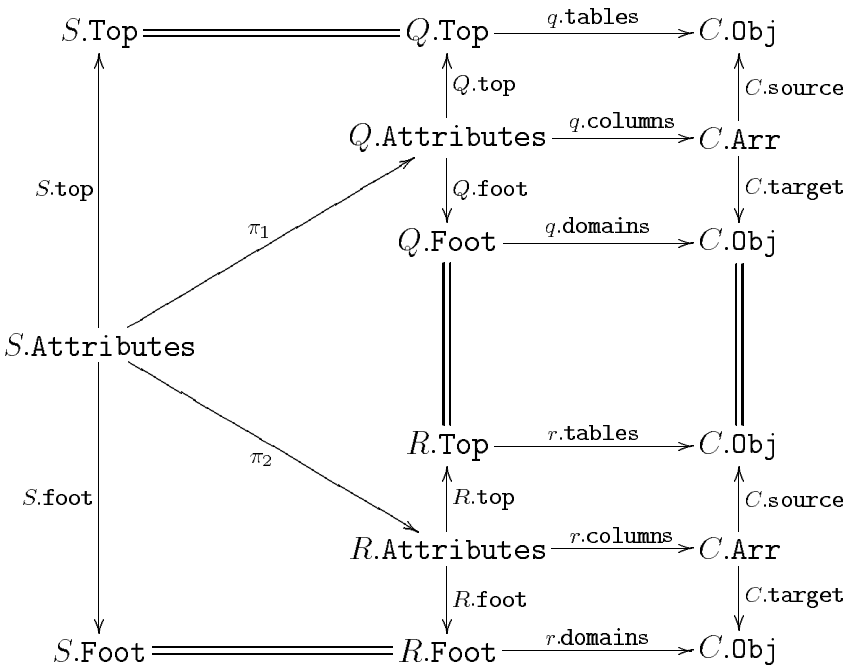
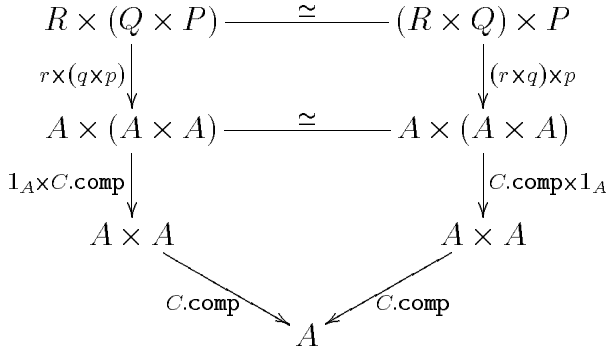


Figure 3: Internal Composition of States

P.Attributes to P, C.Arr to A, and so on, and if we omit the subscripts on $\times$, then the associativity of $\circledast$ reduces to showing the commutativity of this diagram:



The lower part of this diagram is commutative by definition of internal categories, while the top square is commutative by the associativity of the pullback. ∎

There are several ways that the internal composition of states can be used to define a category. For example, one could define a category for which the states are the morphisms, while the objects are the discrete identity states. A more interesting category has the states as its objects and the discrete states as its morphisms:

Definition 24 *The* semantic category of database states *of a schema S in an internal category C, denoted $\mathcal{S}t(S,C)$, is the (external) category defined as follows:*

1. *An object of $\mathcal{S}t(S,C)$ is a database state of S in C.*

2. *For objects s and t of $\mathcal{S}t(S,C)$, a morphism $\phi\colon s \to t$ is a triple (s,f,t) where f is a discrete identity state of $s.\texttt{Top}$ such that $s = t \circledast f$.*

3. *The discrete identity states define the identity morphisms of $\mathcal{S}t(S,C)$, and the composition of morphisms is given by internal composition.*

It is easy to see that if $\phi\colon s \to t$ is a morphism of the category $\mathcal{S}t(S,C)$, then $s.\texttt{domains} = t.\texttt{domains}$. Therefore this category is the disjoint union of the full subcategories $\mathcal{S}t(S,\delta,C)$, whose set of objects is $\{s | s.\texttt{domains} = \delta\}$. In other words, two states can be compared if and only if they have the *same* domains, but they do not need to have the same or even isomorphic tables. This asymmetry between the tops and feet of a schema is an important aspect of database states.

Definition 25 *Two database states s and t are* (semantically) isomorphic *if they are isomorphic in a semantic category. In particular, this implies that they have the same schema, internal category and domains (i.e., $s.\texttt{domains} = t.\texttt{domains}$). A* substate *of a state t is a subobject of t in its semantic category.*

Intuitively, if one relabels the rows of the tables of a state, then one obtains an isomorphic state, provided that the relabeled rows continue to have the same columns as the original elements. From the point of view of database management, this says that one may permute the internal tuple identifiers without changing the semantics of a table. One cannot do the same for the domains of a state. Consider the employee database in the introduction. If the 20-character strings `"John Green"` and `"Ada Byron"` were interchanged everywhere in the tables where this type occurred, then it is intuitively clear that the semantics of the state would be changed.

The morphisms in $\mathcal{S}t(S,C)$ are closely related to the internal analog of the concept of a natural transformation. Let C and D be two internal categories, and let $F, G\colon C \to D$ be two internal functors. An *internal natural transformation* $\tau\colon F \to G$ is a discrete state τ of $C.\texttt{Obj}$ such that $\tau \circledast \mathcal{U}(F) = \mathcal{U}(G) \circledast \tau$. Write $\mathcal{N}at[C,D]$ for the external category of internal functors $F\colon C \to D$ with internal natural transformations as the morphisms and composition defined by $\circledast$.

An internal natural transformation $\tau\colon F \to G$ necessarily satisfies $\tau.\texttt{tables} = F.\texttt{obj}$ and $\tau.\texttt{domains} = G.\texttt{obj}$. Thus τ is completely determined by the morphism $\tau.\texttt{columns}$, and this is how an internal natural transformation is normally defined. In the following, we will use both points of view, and allow the context to determine which is intended.

6 Join.

The join is a fundamental operation in the standard database systems, and the analogous operation is also fundamental in our theory. The standard database concept of a join involves

combining the two relations into a single relation. The surprising aspect of our operation is that it involves just one database state.

Let $m: S \to T$ be a morphism of schemas, and suppose that $s: S \to \mathcal{U}(C)$ is a state of S in C. This is the situation used for defining the extraction operation, except that the morphism of schemas goes the other direction. The join will be a state of T that will be denoted $m_*(s)$. We first give the definition and then follow with a detailed example to illustrate the concept.

Definition 26 *Let $m: S \to T$ be a morphism of schemas, and let s be a state of S in C. Let $U = \mathcal{D}(m.\texttt{tables})$, and let $m = n \circ p$ be the factorization of m given by diagram 7. A* join *of s over m is a state t of T in C such that there is a state u of U in C with the following properties:*

1. *The internal composition $s \circledast u$ is well-defined and coincides with the extraction $n^*(t)$ of t.*
2. *If u' is another state of U such that $s \circledast u'$ is well-defined and coincides with $n^*(t')$ for some state t' of T, then there exists a unique state v of $\mathcal{D}(T.\texttt{Top})$ such that $u \circledast v = u'$.*

Let S be the schema discussed in Example 6. A natural operation for this schema would be to join the two tables to produce a single table. To be precise consider the following SQL query:

```
SELECT E.SSN, E.NAME, E.DEPT, E.ADDR, D.NAME
FROM E EMPLOYEES, D DEPARTMENTS
WHERE E.DEPT = D.CODE
```

We will show that there is a natural morphism of schemas $m: S \to T$ such that for a state s of S, the result of the query is given by $m_*(s)$. Accordingly, first define the schema T as follows:

1. $T.\texttt{Top} = \{ED\}$,
2. $T.\texttt{Attributes} = \{E.SSN, E.NAME, E.DEPT, E.ADDR, D.NAME\}$,
3. $T.\texttt{Foot} = S.\texttt{Foot}$,
4. The morphisms $T.\texttt{top}$ and $T.\texttt{foot}$ have the obvious definitions.

The morphism $m: S \to T$ is given by:

1. $m.\texttt{tables} = \{EMPLOYEES \mapsto ED, DEPARTMENTS \mapsto ED\}$,
2. $m.\texttt{columns} = \{E.SSN \mapsto E.SSN, \ldots, D.CODE \mapsto E.DEPT, \ldots\}$,
3. $m.\texttt{domains} = 1_{S.\texttt{Foot}}$.

The intermediate schema $S \otimes \mathcal{D}(T.\texttt{Top})$ has the same tops as T, but is otherwise the same as S. The morphism m collapses the two top elements of S to one top element of T, and two of the attributes of S are collapsed to a single attribute of T. The factorization $m = n \circ p$ from diagram 7 separates these two parts of the effect of m. p collapses the two top elements of S, while n collapses the attributes of S.

The schema $U = \mathcal{D}(m.\texttt{tables})$ has the following structure:

1. $U.\texttt{Top} = \{ED\}$,
2. $U.\texttt{Attributes} = \{EMPLOYEES, DEPARTMENTS\}$,
3. $U.\texttt{Foot} = U.\texttt{Attributes} = \{EMPLOYEES, DEPARTMENTS\}$,
4. The morphisms $U.\texttt{top}$ and $U.\texttt{foot}$ have the obvious definitions.

A state of U that is compatible with a state s of S can be regarded as a table of rows, each of which is a pair consisting of a row of the $EMPLOYEES$ table and a row of the $DEPARTMENTS$ table. In other words, U is a form of nested relation or, more precisely, a nested table. (See [9] for definitions related to nested relations.) The operation $\circledast$ "unnests" the nested table to produce an ordinary table with all six attributes of S. A state t of T, however, has only five attributes. The condition $n^*(t) = s \circledast u$ constrains $s \circledast u$ to be a state for which the two attributes $E.DEPT$ and $D.CODE$ map to the same column. This is exactly the join condition in the `WHERE` clause.

The only remaining requirement of the join operation is the universality property. It is easy to see that the effect of this condition is to ensure that the state u is the "largest" possible subject to the other conditions. It follows that $m_*(s)$ is precisely the join defined by the SQL query.

Consider some variations of the join. In the first, we select only three of the attributes to be in the final result. The SQL query would then look like the following:

```
SELECT E.NAME, E.ADDR, D.NAME
FROM E EMPLOYEES, D DEPARTMENTS
WHERE E.DEPT = D.CODE
```

One can obtain this result by using the previously discussed join followed by an extraction. Let R be the schema having a single top element, three attributes as in the query and the same feet as in S and T. Define the morphism $r\colon R \to T$ to be inclusion of R as a subschema of T. It is clear that $r^*(m_*(s))$ gives the state corresponding to the query above.

If it happens that two of the rows produced by this query have the same attribute values. In other words, the result of the query need not be a relation. Both the SQL query and the state $r^*(m_*(s))$ agree in this respect. To obtain a relation one must explicitly eliminate duplicates. In SQL one uses the following query:

```
SELECT UNIQUE E.NAME, E.ADDR, D.NAME
FROM E EMPLOYEES, D DEPARTMENTS
```

```
WHERE E.DEPT = D.CODE
```

The database state corresponding to this query is $Sq(r^*(m_*(s)))$, where Sq is the squeeze operation to be defined in section 7.

The main results of this section are that some internal categories have joins, and the characterization of such categories.

Definition 27 *Let D be an internal category in the ambient topos, $\mathcal{A}$. Let X be an object of the ambient topos. Let $F\, D \to \mathcal{D}(X)$ be an internal functor. Recall that the internal hom functor F^* maps $G \in \mathcal{N}at[\mathcal{D}(X), C]$ into $GF.\mathtt{comp} \in \mathcal{N}at[D, C]$. An internal category C is a* complete internal category *iff for each internal category, D, for each object, X, of the ambient topos, $\mathcal{A}$, and for each internal functor $F : D \to \mathcal{D}(X)$, the internal hom functor $F^*\, \mathcal{N}at[\mathcal{D}(X), C] \to \mathcal{N}at[D, C]$ has a right adjoint.*

Theorem 28 *Let $m\colon S \to T$ be an epimorphism of schemas, and let s be a state of S in a complete internal category C, such that $s.\mathtt{domains}$ factors through $m.\mathtt{domains}$. Then there exists a join $t\colon T \to \mathcal{U}(C)$ of s over m.*

Proof. Because m is a state, the following diagram commutes :

$$\begin{array}{ccc}
S.\mathtt{Top} & \xrightarrow{m.\mathtt{tables}} & T.\mathtt{Top} \\
{\scriptstyle S.\mathtt{top}}\uparrow & & \| \\
S.\mathtt{Attributes} & \xrightarrow{m.\mathtt{tables} \circ S.\mathtt{top}} & T.\mathtt{Top} \\
{\scriptstyle m.\mathtt{columns}}\downarrow & & \| \\
T.\mathtt{Attributes} & \xrightarrow{T.\mathtt{top}} & T.\mathtt{Top}
\end{array} \tag{8}$$

We view the first column above as a schema, and define D to be the internal category generated by it. We view the second column as the underlying schema of the discrete internal category $\mathcal{D}(T.\mathtt{Top})$. Therefore the rows of the diagram define an internal functor $F\colon D \to \mathcal{D}(T.\mathtt{Top})$.

Because s is a state, the following diagram commutes:

$$\begin{array}{ccc}
S.\mathtt{Top} & \xrightarrow{s.\mathtt{tables}} & C.\mathtt{Obj} \\
{\scriptstyle S.\mathtt{top}}\uparrow & & \uparrow{\scriptstyle C.\mathtt{source}} \\
S.\mathtt{Attributes} & \xrightarrow{s.\mathtt{columns}} & C.\mathtt{Arr} \\
{\scriptstyle m.\mathtt{columns}}\downarrow & & \downarrow{\scriptstyle C.\mathtt{target}} \\
T.\mathtt{Attributes} & \xrightarrow{\delta \circ T.\mathtt{foot}} & C.\mathtt{Obj}
\end{array} \tag{9}$$

where δ is the unique morphism such that $s.\mathtt{domains} = \delta \circ m.\mathtt{domains}$. It exists because $s.\mathtt{domains}$ factors through $m.\mathtt{domains}$ by hypothesis. It is unique because m is epic.

The first column of diagram 9 is the same as that of diagram 8. Therefore, diagram 9 defines an internal functor $d\colon D \to C$.

Since C is a complete internal category the functor $F^*{:}\mathcal{N}at[\mathcal{D}(T.\mathtt{Top}), C] \to \mathcal{N}at[D, C]$ has a right adjoint $\lim_{\leftarrow F}$. We will use both the unit $\eta{:}\, 1_{\mathcal{N}at[\mathcal{D}(T.\mathtt{Top}),C]} \to \lim_{\leftarrow F} \circ F^*$, and the counit $\epsilon{:}\, F^* \circ \lim_{\leftarrow F} \to 1_{\mathcal{N}at[D,C]}$ of this adjunction.

The limit $l = \lim_{\leftarrow F}(d)$ is an internal functor $l{:}\,\mathcal{D}(T.\mathtt{Top}) \to C$. By definition of $\mathcal{D}$ as an adjoint functor, l is determined by a morphism $\mathcal{V}(l){:}\, T.\mathtt{Top} \to C.\mathtt{Obj}$.

The value of the counit on d is an internal natural transformation $\epsilon(d){:}\, F^*(\lim_{\leftarrow F}(d)) \to d$. Note that $F^*(\lim_{\leftarrow F}(d)) = F^*(l) = l \circ F$. By definition of an internal natural transformation, the following diagram commutes:

$$\begin{array}{ccc}
D.\mathtt{Obj} & \xrightarrow{(l\circ F).\mathtt{obj}} & C.\mathtt{Obj} \\
\| & & \uparrow{\scriptstyle C.\mathtt{source}} \\
D.\mathtt{Obj} & \xrightarrow{\epsilon(d)} & C.\mathtt{Arr} \\
\| & & \downarrow{\scriptstyle C.\mathtt{target}} \\
D.\mathtt{Obj} & \xrightarrow{d.\mathtt{obj}} & C.\mathtt{Obj}
\end{array} \tag{10}$$

As shown in the proof of Theorem 8, $D.\mathtt{Obj}$ can be identified with the coproduct $S.\mathtt{Top} + T.\mathtt{Attributes}$. Now compose the horizontal morphisms in diagram 10 with the inclusion morphism $\iota_2{:}\, T.\mathtt{Attributes} \to D.\mathtt{Obj}$, and substitute the definitions of the top and bottom morphisms on $T.\mathtt{Attributes}$, to obtain the following commutative diagram:

$$\begin{array}{ccccc}
T.\mathtt{Attributes} & \xrightarrow{T.\mathtt{top}} & T.\mathtt{Top} & \xrightarrow{\mathcal{V}(l)} & C.\mathtt{Obj} \\
\| & & & & \uparrow{\scriptstyle C.\mathtt{source}} \\
T.\mathtt{Attributes} & & \xrightarrow{\epsilon(d)\circ\iota_2} & & C.\mathtt{Arr} \\
\| & & & & \downarrow{\scriptstyle C.\mathtt{target}} \\
T.\mathtt{Attributes} & \xrightarrow{T.\mathtt{foot}} & T.\mathtt{Foot} & \xrightarrow{\delta} & C.\mathtt{Obj}
\end{array} \tag{11}$$

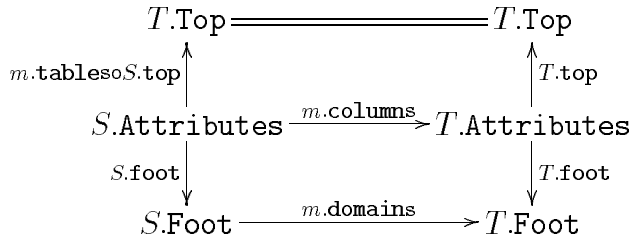
Clearly, diagram 11 defines a state t of T where $t.\mathtt{tables} = \mathcal{V}(l)$, etc. We propose to show that t is a join of s over m.

Next consider the $S.\mathtt{Top}$ component of diagram 10. As before, compose with ι_1 and substitute definitions to obtain the commutative diagram:

$$\begin{array}{ccccc}
S.\mathtt{Top} & \xrightarrow{m.\mathtt{tables}} & T.\mathtt{Top} & \xrightarrow{\mathcal{V}(l)} & C.\mathtt{Obj} \\
\| & & & & \uparrow{\scriptstyle C.\mathtt{source}} \\
S.\mathtt{Top} & & \xrightarrow{\epsilon(d)\circ\iota_1} & & C.\mathtt{Arr} \\
\| & & & & \downarrow{\scriptstyle C.\mathtt{target}} \\
S.\mathtt{Top} & & \xrightarrow{s.\mathtt{tables}} & & C.\mathtt{Obj}
\end{array} \tag{12}$$

It is easy to see that the diagram above defines a state u of the schema $U = \mathcal{F}_t(m.\mathtt{tables})$. More precisely, $u.\mathtt{tables} = \mathcal{V}(l)$, $u.\mathtt{columns} = \epsilon(d) \circ \iota_1$ and $u.\mathtt{domains} = s.\mathtt{tables}$. This last equation implies that u is compatible with s so that $s \circledast u$ is well-defined.

We need to show that $s \circledast u$ coincides with $n^*(t) = t \circ n$, where n is the morphism obtained from the factorization of m given by diagram 7:



First consider $(s \circledast u).\texttt{tables} = u.\texttt{tables} = \mathcal{V}(l) = t.\texttt{tables} = t.\texttt{tables} \circ n.\texttt{tables}$, because $n.\texttt{tables} = 1_{T.\texttt{Top}}$. Similarly, $(s \circledast u).\texttt{domains} = s.\texttt{domains} = \delta \circ m.\texttt{domains} = t.\texttt{domains} \circ m.\texttt{domains} = t.\texttt{domains} \circ n.\texttt{domains}$. The last component is the hardest. Now $(s \circledast u).\texttt{columns} = C.\texttt{comp} \circ \langle \epsilon(d) \circ \iota_1 \circ S.\texttt{top}, s.\texttt{columns} \rangle$, while $t.\texttt{columns} \circ n.\texttt{columns} = \epsilon(d) \circ \iota_2 \circ m.\texttt{columns}$. To relate these two consider the requirement that $\epsilon(d)$ be an internal natural transformation:

$$\epsilon(d) \circledast \mathcal{U}(F^*(\varprojlim_F (d))) = \mathcal{U}(d) \circledast \epsilon(d). \tag{13}$$

The two sides of the equation above are morphisms from $\mathcal{U}(D)$ to $\mathcal{U}(C)$. Consider just the .columns component. This is a morphism $D.\texttt{Arr} \to C.\texttt{Arr}$. Now $D.\texttt{Arr}$ can be identified with $S.\texttt{Top} + S.\texttt{Attributes} + T.\texttt{Attributes}$. Consider just the restriction to $S.\texttt{Attributes}$.

The restriction of the left-hand side of equation 13 is

$$C.\texttt{comp} \circ \langle C.\texttt{id} \circ \mathcal{V}(l) \circ m.\texttt{tables} \circ S.\texttt{top}, \epsilon(d) \circ \iota_2 \circ m.\texttt{columns} \rangle.$$

The left component above can be replaced as follows:

$$\begin{aligned} & C.\texttt{id} \circ \mathcal{V}(l) \circ m.\texttt{tables} \circ S.\texttt{top} \\ &= C.\texttt{id} \circ \mathcal{V}(l) \circ T.\texttt{top} \circ m.\texttt{columns}, \text{ since } m \text{ is a morphism,} \\ &= C.\texttt{id} \circ C.\texttt{source} \circ \epsilon(d) \circ \iota_2 \circ m.\texttt{columns}, \text{ by diagram 11} \end{aligned}$$

Using this fact and the definition of an internal category, gives the following for the restriction of the left-hand side of equation 13:

$$\begin{aligned} & C.\texttt{comp} \circ \langle C.\texttt{id} \circ C.\texttt{source} \circ \epsilon(d) \circ \iota_2 \circ m.\texttt{columns}, \epsilon(d) \circ \iota_2 \circ m.\texttt{columns} \rangle \\ &= C.\texttt{comp} \circ \langle C.\texttt{id} \circ C.\texttt{source}, 1_{C.\texttt{Arr}} \rangle \circ \epsilon(d) \circ \iota_2 \circ m.\texttt{columns} \\ &= \epsilon(d) \circ \iota_2 \circ m.\texttt{columns} \\ &= t.\texttt{columns} \circ n.\texttt{columns}, \text{ as noted earlier.} \end{aligned}$$

On the other hand, the restriction of the right-hand side of equation 13 is:

$$C.\texttt{comp} \circ \langle \epsilon(d) \circ \iota_1 \circ S.\texttt{top}, s.\texttt{columns} \rangle = (s \circledast u).\texttt{columns}.$$

Thus $s \circledast u = t \circ n$ as desired. It follows that t and u satisfy the first requirement for being a join.

Now let t' and u' be a pair of states satisfying this requirement. The morphism

$$t'.\texttt{tables}: T.\texttt{Top} \to C.\texttt{Obj}$$

defines an internal functor

$$l' = \mathcal{D}(t'.\texttt{tables}): \mathcal{D}(T.\texttt{Top}) \to C.$$

Applying the unit η of the adjunction between F^* and $\varprojlim_F$ to l' yields an internal natural transformation $\eta(l'): l' \to \varprojlim_F(F^*(l'))$.

Now let $g = [u'.\texttt{columns}, t'.\texttt{columns}]: D.\texttt{Obj} = S.\texttt{Top} + T.\texttt{Attributes} \to C.\texttt{Arr}$. We now show that g defines an internal natural transformation $g: F^*(l') \to d$. As a discrete state g is given by the following diagram:

$$\begin{array}{ccc}
D.\texttt{Obj} & \xrightarrow{l'\circ[m.\texttt{tables}, T.\texttt{top}]} & C.\texttt{Obj} \\
\| & & \uparrow{\scriptstyle C.\texttt{source}} \\
D.\texttt{Obj} & \xrightarrow{[u'.\texttt{columns}, t'.\texttt{columns}]} & C.\texttt{Arr} \\
\| & & \downarrow{\scriptstyle C.\texttt{target}} \\
D.\texttt{Obj} & \xrightarrow{[s.\texttt{tables}, \delta\circ T.\texttt{foot}]} & C.\texttt{Obj}
\end{array}$$

The first component of this diagram commutes because u' is a state, and the second commutes because t' is a state. The requirement for g to be an internal natural transformation is $g \circledast \mathcal{U}(F^*(l')) = \mathcal{U}(d) \circledast g$. The only nontrivial component of this equation is the .columns component. This is a morphism $D.\texttt{Arr} \to C.\texttt{Arr}$. Now $D.\texttt{Arr}$ has three components. On two of the components, the requirement is a consequence of condition (d) of the definition of an internal category. On the third component, $S.\texttt{Attributes}$, the requirement is a consequence of the hypothesis $s \circledast u' = n^*(t')$.

Applying the functor $\varprojlim_F$ to g produces an internal natural transformation

$$\varprojlim_F(g): \varprojlim_F(F^*(l')) \to l = \varprojlim_F(d)$$

. Composing this internal natural transformation with $\eta(l')$ gives a discrete state $v = \varprojlim_F(g) \circledast \eta(l')$ of $T.\texttt{Top}$, which is also an internal natural transformation $v: l' \to l$.

We claim that $u \circledast v = u'$. It's easy to see that $(u \circledast v).\texttt{tables} = v.\texttt{tables} = t'.\texttt{tables} = u'.\texttt{tables}$ because $s \circledast u' = t' \circ n$. It is also easy to check that $(u \circledast v).\texttt{domains} = u.\texttt{domains} = u'.\texttt{domains}$ because both u and u' are compatible with s. It is more difficult to verify that $(u \circledast v).\texttt{columns} = u'.\texttt{columns}$. To see this, expand:

$$\begin{aligned}
&(u \circledast v).\texttt{columns} \\
&\quad = C.\texttt{comp} \circ \langle v.\texttt{columns} \circ m.\texttt{tables}, \epsilon(d) \circ \iota_1 \rangle \\
&\quad = C.\texttt{comp} \circ \langle v.\texttt{columns} \circ F \circ \iota_1, \epsilon(d) \circ \iota_1 \rangle \\
&\quad = C.\texttt{comp} \circ \langle v.\texttt{columns} \circ F, \epsilon(d) \rangle \circ \iota_1 \\
&\quad = (\epsilon(d) \circledast (v \circ F)).\texttt{columns} \circ \iota_1 \\
&\quad = (\epsilon(d) \circledast (F^*(v))).\texttt{columns} \circ \iota_1
\end{aligned}$$

We now compute the internal composition above:

$$\begin{aligned}
\epsilon(d)\circledast F^*(v) &= \epsilon(d)\circledast F^*(\varprojlim_F(g)\circledast\eta(l')) \\
&= \epsilon(d)\circledast F^*(\varprojlim_F(g))\circledast F^*(\eta(l')), \text{ by Prop 21} \\
&= \epsilon(d)\circledast(F^*\circ\varprojlim_F)(g)\circledast F^*(\eta(l')) \\
&= g\circledast\epsilon(F^*(l'))\circledast F^*(\eta(l')), \text{ since } \epsilon \text{ is an external natural transformation} \\
&= g\circledast 1_{F^*(l')}, \text{ by one of the formulas relating the unit and counit} \\
&= g
\end{aligned}$$

Substituting this result into the earlier one, we obtain: $(u\circledast v).\texttt{columns} = g.\texttt{columns}\circ\iota_1 = u'.\texttt{columns}$. Therefore, $uv.\texttt{comp} = u'$ as desired.

It remains only to show that v is unique with respect to the condition $u\circledast v = u'$. Let v' be another discrete state of $T.\texttt{Top}$ satisfying this condition. It is immediate that $v.\texttt{tables} = v'.\texttt{tables}$ and $v.\texttt{domains} = v'.\texttt{domains}$. Now by definition, $v.\texttt{columns} = (\varprojlim_F(g)\circledast\eta(l')).\texttt{columns}$, where $g = [u'.\texttt{columns}, t'.\texttt{columns}]$.

We compute each of the components of g. First, consider $u'.\texttt{columns}$. By the condition on v',

$$\begin{aligned}
u'.\texttt{columns} &= (u\circledast v').\texttt{columns} \\
&= C.\texttt{comp}\circ\langle v'.\texttt{columns}\circ m.\texttt{tables}, \epsilon(d)\circ\iota_1\rangle \\
&= C.\texttt{comp}\circ\langle v'.\texttt{columns}\circ F\circ\iota_1, \epsilon(d)\circ\iota_1\rangle \\
&= C.\texttt{comp}\circ\langle v'.\texttt{columns}\circ F, \epsilon(d)\rangle\circ\iota_1 \\
&= F^*(v'.\texttt{columns})\circledast\epsilon(d).\texttt{columns}\circ\iota_1
\end{aligned}$$

Next, compute $t'.\texttt{columns}$. By the assumption that t' and u' define a join, $t'\circ n = s\circledast u' = s\circledast u\circledast v' = (t\circ n)\circledast v'$. Now expand the .columns component to obtain:

$$\begin{aligned}
&((t\circ n)\circledast v').\texttt{columns} \\
&= C.\texttt{comp}\circ\langle v'.\texttt{columns}\circ m.\texttt{tables}\circ S.\texttt{top}, t.\texttt{columns}\circ n.\texttt{columns}\rangle \\
&= C.\texttt{comp}\circ\langle v'.\texttt{columns}\circ T.\texttt{top}\circ m.\texttt{columns}, t.\texttt{columns}\circ m.\texttt{columns}\rangle \\
&\qquad\qquad \text{since } m \text{ is a state and by definition of } n \\
&= C.\texttt{comp}\circ\langle v'.\texttt{columns}\circ T.\texttt{top}, t.\texttt{columns}\rangle\circ m.\texttt{columns} \\
&= C.\texttt{comp}\circ\langle v'.\texttt{columns}\circ F\circ\iota_2, \epsilon(d)\iota_2\rangle\circ m.\texttt{columns} \\
&\qquad\qquad \text{by the definitions of } F \text{ and } t \\
&= C.\texttt{comp}\circ\langle v'.\texttt{columns}\circ F, \epsilon(d)\rangle\circ\iota_2\circ m.\texttt{columns} \\
&= (\epsilon(d)\circledast F^*(v'.\texttt{columns})).\texttt{columns}\circ\iota_2\circ m.\texttt{columns}
\end{aligned}$$

This is equal to

$$(t'\circ n).\texttt{columns} = t'.\texttt{columns}\circ n.\texttt{columns} = t'.\texttt{columns}\circ m.\texttt{columns}.$$

Since $m.\texttt{columns}$ is an epimorphism, it follows that

$$t'.\texttt{columns} = (\epsilon(d) \circledast F^*(v'.\texttt{columns})).\texttt{columns} \circ \iota_2.$$

Comparing this with the computation of $u'.\texttt{columns}$, we see that

$$g = (\epsilon(d) \circledast F^*(v'.\texttt{columns})).\texttt{columns}.$$

Regarding g as an internal natural transformation we then have:

$$g = \epsilon(d) \circledast F^*(v'.\texttt{columns}).$$

Now expand the definition of v as follows:

$$\begin{aligned}
v.\texttt{columns} &= (\varprojlim_F (g) \circledast \eta(l')).\texttt{columns} \\
&= (\varprojlim_F (\epsilon(d) \circledast F^*(v'.\texttt{columns})) \circledast \eta(l')).\texttt{columns} \\
&= (\varprojlim_F \epsilon(d) \circledast \varprojlim_F (F^*(v'.\texttt{columns})) \circledast \eta(l')).\texttt{columns}, \\
&\qquad\qquad \text{since } \varprojlim_F \text{ is a functor} \\
&= (\varprojlim_F \epsilon(d) \circledast \eta(l) \circledast v'.\texttt{columns}).\texttt{columns}, \\
&\qquad\qquad \text{since } \eta \text{ is an external natural transformation} \\
&= (\varprojlim_F \epsilon(d) \circledast \eta(\varprojlim_F (d)) \circledast v'.\texttt{columns}).\texttt{columns} \\
&= (1_{\varprojlim_F (d)} \circledast v'.\texttt{columns}).\texttt{columns}, \\
&\qquad\qquad \text{by one of the formulas satisfied by the unit and counit} \\
&= v'.\texttt{columns}
\end{aligned}$$

Therefore $v = v'$ and the result follows. ∎

As defined, it is not obvious that the join is unique in any sense. The following result shows that in the usual case where m is an epimorphism of schemas the join is unique up to isomorphism.

Theorem 29 *Let $m: S \to T$ be a morphism of schemas, and let s be a state of S in C. Let t and t' be two states of T that are joins of s over m. Then there exists a discrete isomorphism state v of $\mathcal{D}(T.\texttt{Top})$ such that $t = t' \circledast v$.*

Proof. Let $m = n \circ p$ be the factorization of m given by diagram 7. Let R and U be the discrete schemas $\mathcal{D}(T.\texttt{Top})$ and $\mathcal{D}(m.\texttt{tables})$, respectively.

By definition of a join, there are states u and u' of U such that $n^*(t) = s \circledast u$ and $n^*(t') = s \circledast u'$. In addition, there are unique states v and v' such that $u' \circledast v = u$ and $u \circledast v' = u'$, respectively. By the usual reasoning, we find that $v \circledast v'$ and $v' \circledast v$ are discrete identity states. Therefore both v and v' are discrete isomorphism states.

We now compute $n^*(t)$ in terms of t' and v as follows:

$$
\begin{array}{lcll}
n^*(t) & = & s \circledast u & \text{By construction} \\
& = & s \circledast (u' \circledast v) & \text{By the choice of } v \\
& = & (s \circledast u') \circledast v & \text{By Theorem 23} \\
& = & n^*(t') \circledast v & \text{By construction} \\
& = & (t' \circ n) \circledast v & \text{By definition of extraction} \\
& = & (t' \circ n) \circledast (v \circ 1_R) & \text{Note that } n.\texttt{tables} = 1_{T.\texttt{Top}} \\
& & & \quad \text{so } 1_R \text{ is compatible with } n. \\
& = & C.\texttt{comp} \circ ((t' \circ n) \otimes (v \circ 1_R)) & \text{By definition of } \circledast \\
& = & C.\texttt{comp} \circ ((t' \otimes v) \circ (n \otimes 1_R)) & \text{By Proposition 21} \\
& = & (C.\texttt{comp} \circ (t' \otimes v)) \circ n & \text{By associativity of } \circ \\
& = & (t' \circledast v) \circ n & \text{By definition of } \circledast
\end{array}
$$

Therefore $t \circ n = (t' \circledast v) \circ n$. Now m is an epic morphism, and the components of m coincide with those of n except for $n.\texttt{tables}$ which is an identity morphism. Therefore n is also epic. It follows that $t = t' \circledast v$. The result then follows. ∎

Having shown that join is unique up to isomorphism, one should also check that isomorphic states produce isomorphic joins in general.

Theorem 30 *Let $m: S \to T$ be an epimorphism of schemas, and let s, s' be isomorphic states of S. Then $m_*(s)$ is isomorphic to $m_*(s')$.*

Proof. We will use the notation in the proof of Theorem 29. Since s is isomorphic to s', there exists a discrete isomorphism state w such that $s \circledast w = s'$. By definition of the join, there are states u and u' on the schema U and states t and t' on the schema T such that $s \circledast u = n^*(t)$ and $s' \circledast u' = n^*(t')$. By the associativity of $\circledast$, $n^*(t') = s' \circledast u' = (s \circledast w) \circledast u' = s \circledast (w \circledast u')$. By the universal property defining the join state t, there exists a unique discrete state v such that $w \circledast u' \circledast v = u$. Since w is a discrete isomorphism state, it has an inverse w'. Now repeat the argument with the roles of t and t' reversed. The result is a unique discrete state v' such that $w' \circledast u \circledast v' = u'$. Substituting each of these into the other gives $u' \circledast v \circledast v' = u'$ and $u \circledast v' \circledast v = u$. By the universal property of u and u', it follows that v and v' are discrete isomorphism states. Furthermore, $u \circledast v' = w \circledast u'$ and $u' \circledast v = w' \circledast u$.

We now compute $t' \circ n = n^*(t') = s' \circledast u' = s \circledast w \circledast u' = s \circledast u \circledast v' = n^*(t) \circledast v' = (t \circ n) \circledast v'$. As in the proof of Theorem 29, $(t \circ n) \circledast v' = (t \circledast v') \circ n$. Since n is epic, it follows that $t' = t \circledast v$. The result then follows. ∎

7 Squeezing and Indexing Tables.

So far the tables of a database state are, in general, multisets not relations. It is possible for two rows of a table to have exactly the same attributes, and yet be different rows. In this section, a condition is introduced that constrains a state to be relational, and an operation is defined that converts an arbitrary state to a relation. The operation is called *squeezing* and it corresponds to the "elimination of duplicates" operation of standard database systems.

Before giving the precise definition of relations and squeezing, we discuss a closely related concept and a categorical formulation of it.

In practice, the elimination of duplicates is done by creating an auxiliary table rather than modifying the original table. If this auxiliary table retains links to the original tuples, then it is called an *index*.[3] Indexing is important for improving the performance of database queries. To see how one can express indexing in categorical terms, consider the case of a relation R one of whose attributes is a function $a: R \to D$ with values in a domain D. An index for this attribute is a set whose elements correspond to some of the elements of D, i.e., there is a monic function $m: I \to D$. In addition, each of the original tuples of R is linked with its index entry. This is most easily expressed as an epic function $e: R \to I$ such that $a = m \circ e$. In other words, an index of an attribute is an "epic-monic factorization" of the attribute as a function. This leads to the following definition:

Definition 31 *An* indexable internal category *is an internal category with monic arrows and epic arrows together with a morphism* $C.\mathtt{index}: C.\mathtt{Arr} \to C.\mathtt{Mon} \times_{C.\mathtt{Obj}} C.\mathtt{Epi}$ *such that*

$$C.\mathtt{comp} \circ \langle C.\mathtt{mon} \circ \pi_1, C.\mathtt{epi} \circ \pi_2 \rangle \circ C.\mathtt{index} = 1_{C.\mathtt{Arr}}.$$

While an epic-monic factorization can be used to express the concept of an index, it is not enough by itself for eliminating duplicates in general. It is also necessary to choose one out of each set of duplicates. Such a choice is arbitrary and so is an example of a use of the axiom of choice. In addition, a table will generally have several attributes. To express these as a single function, one must construct the product of the domain sets as well as the product of the attribute functions. This construction is a special case of limits.

Having introduced the background needed by the squeeze operation, we now give the main concepts and results of this section.

Definition 32 *Let s be a state of a schema S in an internal category C. The state s is said to be* relational *(or a* monic state*) if for any two discrete states t and u of $S.\mathtt{Top}$, $s \circledast t = s \circledast u$ implies $t = u$. A* relation *is an isomorphism class of relational states of S.*

If $S.\mathtt{Top}$ is a final object of $\mathcal{A}$, then a state can be regarded as a "cone" of arrows having a common source. A monic state of such a schema is essentially the same as a "monic family" as defined in Freyd and Scedrov [4]. Furthermore a relational state of this schema essentially coincides with their concept of a "relation." The only difference is that our concepts of monic and relational states are defined within the context of an ambient topos.

Given an arbitrary state, one should be able to "eliminate duplicates" to obtain a relational state. The resulting operation is called "squeezing."

Definition 33 *Let s be a state of a schema S. A state r of S is said to be a* squeeze *of s if the following hold:*

[3] One should be careful not to confuse this concept of index with the concept of indexed categories or with indexing in mathematics in general. For example, the mathematical concept is an *a priori* structure while the database concept is *a posteriori*.

1. *The state r is monic.*

2. *There exists a discrete state q of $S.\texttt{Top}$ such that $r \circledast q = s$.*

3. *If r' is another state satisfying conditions (1) and (2), then there exists a discrete state t of $S.\texttt{Top}$ such that $r' \circledast t = r$.*

Note that the discrete state q in condition (2) is unique if it exists by definition of a monic state. Similarly, the discrete state t is unique if it exists. Furthermore, if q' is the discrete state for which $r' \circledast q' = s$, then $t \circledast q = q'$. These uniqueness properties imply that a squeeze of a state is unique up to isomorphism. Therefore the isomorphism class of a squeeze of a state is uniquely determined by a state (if any squeezes exist at all) and will be denoted $Sq(s)$.

The main result of this section is a set of conditions on an internal category that ensures that every database state can be squeezed.

Theorem 34 *Let C be an complete, indexable, internal axiom of choice category in an ambient topos $\mathcal{A}$. Then for any schema S and state s of S, there exists a squeeze of s.*

Proof. Let D be the discrete internal category $\mathcal{D}(S.\texttt{Attributes})$, and let d be the internal functor determined by the morphism

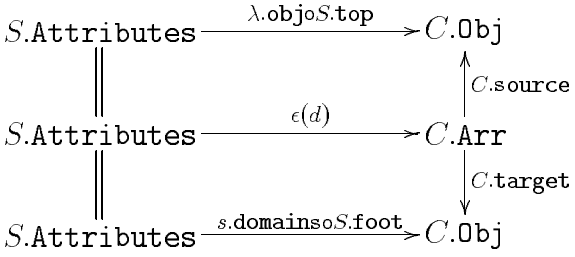
$$C.\texttt{id} \circ s.\texttt{domains} \circ S.\texttt{foot} : S.\texttt{Attributes} \to C.\texttt{Arr}.$$

Let $F : D \to \mathcal{D}(S.\texttt{Top})$ be the internal functor determined by the morphism $S.\texttt{top}$. Since C is complete, the functor

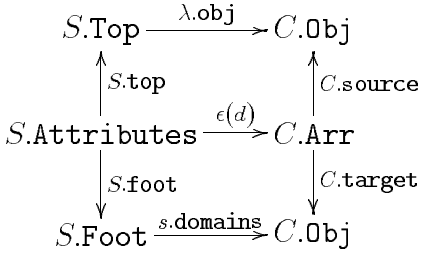
$$F^* : \mathcal{N}at[\mathcal{D}(S.\texttt{Top}), C] \to \mathcal{N}at[D, C]$$

has a right adjoint $\varprojlim_F$. The counit of this adjunction will be denoted ϵ. Let λ be the limit $\varprojlim_F(d)$. Intuitively, the morphism $\lambda.\texttt{obj} : S.\texttt{Top} \to C.\texttt{Obj}$ maps each top in $S.\texttt{Top}$ to the product of the domains of its attributes.

Applying the counit to d gives an internal natural transformation $\epsilon(d)$ from $(F^* \circ \varprojlim_F)(d)$ to d. Therefore, $\epsilon(d) : S.\texttt{Attributes} \to C.\texttt{Arr}$ and the following diagram commutes:

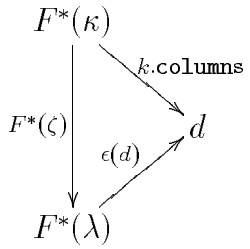


But this is equivalent to the following diagram:



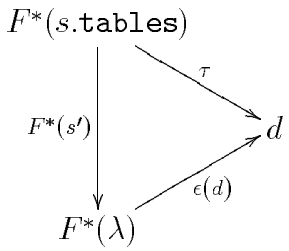
This diagram defines a state of S that will be denoted p. Note that p depends only on the morphism $s.\texttt{domains}$ of s. Intuitively, p replaces each table of $s.\texttt{tables}$ by the cartesian product of the domains of the table, and each column of $s.\texttt{columns}$ is replaced by a projection arrow.

We claim that p is a monic state. Let f and g be discrete states of $S.\texttt{Top}$ such that $p \circledast f = p \circledast g$. Let $k = p \circledast f$. By reversing the procedure above for constructing p from $\epsilon(d)$, one can use the state k to define an internal natural transformation from the internal functor κ determined by $C.\texttt{id} \circ k.\texttt{tables} \circ S.\texttt{top}$. By definition of the adjoint, there is a unique internal natural transformation ζ from κ to λ such that this diagram commutes:



Commutativity of this diagram is equivalent to the condition $p \circledast z = k$, where z is the state corresponding to ζ. Now f and g also satisfy this condition. By the uniqueness of ζ it follows that $f = z = g$, and therefore that p is monic.

Now consider the internal natural transformation τ from $F^*(s.\texttt{tables})$ to d determined by the morphism $s.\texttt{columns}\colon S.\texttt{Attributes} \to C.\texttt{Arr}$, where $s.\texttt{tables}$ denotes the discrete identity state defined by $s.\texttt{tables}$. By definition of the adjoint, there is a unique internal natural transformation s' from $s.\texttt{tables}$ to λ such that this diagram commutes:



Commutativity of this diagram is equivalent to the condition $p \circledast s' = s$. Thus s' is uniquely determined by this condition. Intuitively, for each top of $S.\texttt{Top}$, s' is the product arrow from the table to the cartesian product of the domains.

Now use the fact that C is indexable to define discrete states m and q of $S.\mathtt{Top}$ by

$$m.\mathtt{columns} = C.\mathtt{mon} \circ \pi_1 \circ C.\mathtt{index} \circ s'.\mathtt{columns}$$

$$q.\mathtt{columns} = C.\mathtt{epi} \circ \pi_2 \circ C.\mathtt{index} \circ s'.\mathtt{columns}$$

It follows that $s' = m \circledast q$ and hence that $s = p \circledast m \circledast q$. Let $r = p \circledast m$. It is easy to check that m is a monic state. Since p has already been shown to be a monic state, it follows that r is a monic state. Furthermore, $r \circledast q = p \circledast m \circledast q = s$. Therefore the first two conditions of Definition 33 hold.

It remains to show the uniqueness condition for r. Let r' be another state of S satisfying the first two conditions of Definition 33. Suppose that q' is a discrete state of $S.\mathtt{Top}$ such that $r' \circledast q' = s$. Now use the axiom of choice to define a discrete state h by

$$h.\mathtt{columns} = C.\mathtt{choice} \circ \pi_2 \circ C.\mathtt{index} \circ q.\mathtt{columns},$$

and set $t = q' \circledast h$. Now by definition of $C.\mathtt{choice}$, it follows that $q \circledast h$ is a discrete identity state. Therefore $r' \circledast t = r' \circledast q' \circledast h = s \circledast h = r \circledast q \circledast h = r$. ∎

It would be interesting to determine whether the axiom of choice is essential here. From a database point of view, it is just an implementation issue whether one eliminates duplicates by choosing one of them or by using some other technique.

Intuitively, Theorem 34 proves that the squeeze of any state is a substate of the cartesian product of the domains. The concept of a relation in the relational model is often informally defined to be a subset of a cartesian product of domains. The theorem therefore gives some justification for this as the definition of a relation. To fully justify it we need that every substate of the product state p defined in the proof of Theorem 34 is monic. In fact, it is easy to check that a substate of a monic state is also monic. Therefore under the assumptions of the theorem, relations are precisely the subobjects of the cartesian product states p.

8 Selection.

The selection operation in database management is often the first operation introduced. Indeed many file processing systems will offer elaborate selection operations without offering join or squeeze. However, from a theoretical point of view, the selection operation is more complex because it introduces a new kind of concept entirely: a predicate on values of variables. The kinds of predicates can be very elaborate, involving not only constants, equality and boolean operations, but also ranges, "wild-card matches" and domain-specific operations (such as time and date comparisons). It would carry us far beyond the context of this paper to try to formalize all of this. Instead, we note that however complex a predicate may be, there is always a database state that corresponds to the set of all solutions to the predicate.[4]

[4]The resulting state may not be finite, but this is not a problem in this framework.

Definition 35 *A* predicate state *p of a schema S in an internal category C consists of a morphism of schemas $p.\mathtt{sub}\colon S' \to S$ and a state $p.\mathtt{state}\colon S' \to \mathcal{U}(C)$. A* selection state *of a state s of S with respect to the predicate p is a state t of S satisfying the following two conditions:*

1. *There exists a pair of discrete states u of $S.\mathtt{Top}$ and v of $S'.\mathtt{Top}$ such that $t = s \circledast u$ and $p.\mathtt{sub}^* t = p.\mathtt{state} \circledast v$.*

2. *If u' and v' are a pair of states satisfying*

$$p.\mathtt{state} \circledast v' = p.\mathtt{sub}^*(s \circledast u'),$$

then there exists a unique discrete state w of $S.\mathtt{Top}$ such that

$$u \circledast v = u' \text{ and } v \circledast p.\mathtt{sub}.\mathtt{tables}^* w = v'.$$

Intuitively, the selection state t is "contained" within both the predicate and the original state. The morphism $p.\mathtt{sub}$ is the *substitution* morphism that defines how the tops, attributes and feet of the predicate schema S' are to be substituted by corresponding tops, attributes and feet of S. Note that in practice, the state $p.\mathtt{state}$ is a monic state[5]. It is easy to check that a selection is unique up to isomorphism of states and that isomorphic states have isomorphic selection states.

Example 36 *Return to the employee database in Example 13. We describe how to express the following SQL selection query in this framework:*

```
SELECT *
FROM E EMPLOYEES
WHERE E.NAME = "John Green"
```

Let T be the schema obtained from the employee schema S by extracting just the $EMPLOYEES$ table and its attributes. Define a predicate state p of T as follows. Let T' be the discrete schema on the final object of the ambient topos. Let $p.\mathtt{sub}\colon T' \to T$ be given by

1. The morphism $p.\mathtt{sub}.\mathtt{tables}\colon 1 \to \{EMPLOYEES\}$ maps the single element of 1 to $EMPLOYEES$.

2. The morphism $p.\mathtt{sub}.\mathtt{columns}\colon 1 \to \{SSN, NAME, \ldots\}$ maps the single element of 1 to $NAME$.

3. The morphism $p.\mathtt{sub}.\mathtt{domains}\colon 1 \to \{INTEGER, CHAR(11), CHAR(20), \ldots\}$ maps the single element of 1 to $CHAR(20)$.

[5]When p.`state` is not monic, the selection is still a well defined operation. However, a selection with respect to a nonmonic predicate does not correspond to what one intuitively thinks of as being a "selection".

Let $p.\mathtt{state}\colon T' \to \mathcal{U}(C)$ be given by

1. The morphism $p.\mathtt{state.tables}\colon 1 \to C.\mathtt{Obj}$ maps the single element of 1 to a singleton set $\{0\}$ in $C.\mathtt{Obj}$.

2. The morphism $p.\mathtt{state.columns}\colon 1 \to C.\mathtt{Arr}$ maps the single element of 1 to the arrow $\{0 \mapsto \mathtt{"John\ Green"}\}$.

3. The morphism $p.\mathtt{state.domains}\colon 1 \to C.\mathtt{Obj}$ maps the single element of 1 to the set of 20-character strings.

The selection state t of T consists of those tuples of the $EMPLOYEE$ relation that have NAME equal to "John Green", i.e., the result of the SQL query above.

The main result of this section is that the selection can be expressed in terms of the join.

Theorem 37 *Let s be a state of a schema S in an internal category C, and let p be a predicate of the schema S in C. Let j be the morphism of schemas $[p.\mathtt{sub}, 1_S]\colon S' + S \to S$. Then $j_*[p.\mathtt{state}, s]$, if it exists, is a selection of s with respect to p.*

Proof. Let r be a join state $j_*[p.\mathtt{state}, s]$, and let y denote the database state denoted u in Definition 26. Then $y.\mathtt{columns}\colon S'.\mathtt{Top} + S.\mathtt{Top} \to C.\mathtt{Arr}$. Let u be the discrete state of $S.\mathtt{Top}$ defined by $y \circ \iota_2$, let v be the discrete state of $S'.\mathtt{Top}$ defined by $y \circ \iota_1$, and let $t = s \circledast u$.

We claim that t is a selection of s with respect to p. We have that $t = s \circledast u$ by definition, while the condition $p.\mathtt{sub}^* t = p.\mathtt{state} \circledast v$ follows from the fact that $s \circledast y = n^* r$. The difficult part is the uniqueness property.

Suppose that u' and v' are discrete states satisfying $p.\mathtt{state} \circledast v' = p.\mathtt{sub}^*(s \circledast u')$. Restricting this equation to each component gives the following equations:

$$\begin{aligned} C.\mathtt{source} \circ v' &= C.\mathtt{source} \circ u' \circ p.\mathtt{sub.tables} \\ C.\mathtt{comp} \circ \langle p.\mathtt{state.columns}, v' \circ S'.\mathtt{top} \rangle &= \\ C.\mathtt{comp} \circ \langle s.\mathtt{columns}, u' \circ S.\mathtt{top} \rangle &\circ p.\mathtt{sub.columns} \\ p.\mathtt{state.domains} &= s.\mathtt{domains} \circ p.\mathtt{sub.domains} \end{aligned}$$

Let U and n be the same as in the statement of Theorem 28. Let y' be the state of U given by $y'.\mathtt{tables} = C.\mathtt{source} \circ u'$, $y'.\mathtt{columns} = [v', u']$ and $y'.\mathtt{domains} = C.\mathtt{target} \circ [v', u']$. The first equation in the list above implies that y' is a database state. Now define a state r' of S by $r'.\mathtt{tables} = C.\mathtt{source} \circ u'$, $r'.\mathtt{columns} = C.\mathtt{comp} \circ \langle s.\mathtt{columns}, u' \circ S.\mathtt{top} \rangle$ and $r'.\mathtt{domains} = s.\mathtt{domains}$. The second and third equations in the list above imply that r' satisfies the equation $[p.\mathtt{state}, s] \circledast y' = n^* r'$. By definition of the join, there is a unique discrete state w of S.Top such that $y \circledast w = y'$. Restricting this equation to $S.\mathtt{Top}$ gives $u \circledast w = u'$, while restricting it to $S'.\mathtt{Top}$ gives $v \circledast w.\mathtt{sub.tables}^* w = v'$. It follows that t is a selection as desired. ∎

Corollary 38 *Let s be a state of a schema S in a complete internal category C, and let p be a predicate of the schema S in C. If $p.\mathtt{state.domains} = s.\mathtt{domains} \circ p.\mathtt{sub.domains}$, then there exists a selection of s with respect to p.*

Proof. It is easy to see that the morphism of schemas j in Theorem 37 is an epimorphism. The condition on p implies that

$$[p.\mathtt{state}, s].\mathtt{domains} = s.\mathtt{domains} \circ j.\mathtt{domains}.$$

Since C is complete, Theorem 28 applies and the join $j_*[p.\mathtt{state}, s]$ exists. By Theorem 37, the join state is a selection state for s with respect to p. ∎

9 Boolean Operations.

The only remaining operations of the relational algebra are the boolean operations of intersection, union and difference. They are also the most problematic in this framework since topoi are not in general boolean. Using Definition 25 one can speak of a substate of any database state t. Moreover since substates are just subobjects, there is a partial order on the substates of t. Under some conditions the order algebra formed by the set of substates will be a boolean, or at least, a Heyting algebra. It is an open question what these conditions are in general.

By theorem 34, all monic states are substates of the maximal product state constructed in theorem 34, and the squeeze of any state is monic. In the classical relational case, where all states are monic, this means that all states are substates of the maximal product state. Since the internal category is complete, it seems clear that intersections and unions of states exist, and are the familiar intersections and unions. Turning the phrase "it seems clear" into a theorem is an interesting open question.

10 Conclusion and Future Work.

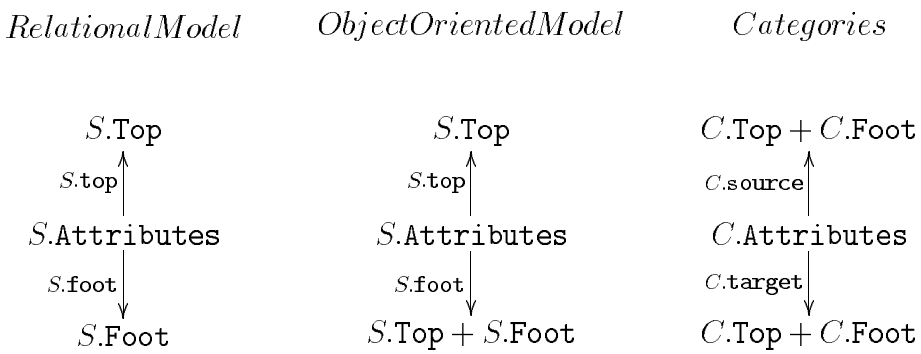
In a database system there are a number of notions:

- attributes
- tables
- domains
- rows of tables
- columns of tables
- elements of domains

Consider for the moment that each of these are simply "things" or "objects." So by "table" we mean "table-as-object" or to be less abstract, think of "table" as being the name of the table.

In a standard database, these are all different objects, and they should all inhabit different universes. Less abstractly, one would say that they all come from different "name spaces." Thus even if by chance a table is named "John Smith," which happens to be the name attribute of one of its records, one nevertheless recognizes that, as an object, the table is quite distinct from the element of the domain of the attribute.

For an object-oriented model, it is no longer the case that all the notions above are disjoint. A domain could also be a table. In a full categorical framework there is no distinction between table and domain. The following diagram summarized the similarities and differences between the three regimes.



What is not shown in the diagram is that the relational model has no useful notion of composition, whereas the other two do. The concept of object-oriented data model sketched above includes concepts of identity, classes, composition of attributes and inheritance. It lacks only an explanation of behavior. It is interesting, however, to note that there is a clear progression from the classical relational model to full category theory.

References

[1] A. Asperti and G. Longo. *Categories, Types, and Structures.* The MIT Press, Cambridge, MA, 1991.

[2] M. Barr and C. Wells. *Category Theory for Computing Science.* Prentice-Hall, Englewood Cliffs, NJ, 1990.

[3] P. Freyd. Aspects of topoi. *Bull. Australian Math. Soc.*, 7:1–76, 1972.

[4] P. Freyd and A. Scedrov. *Categories, Allegories.* North-Holland, Amsterdam, 1990.

[5] R. Goldblatt. *Topoi – The Categorical Analysis of Logic*, volume 98. North-Holland, Amsterdam, 1979.

[6] J.G. Hughes. *Database Technology.* Prentice Hall, New York, 1988. A software engineering approach.

[7] P. Johnstone. *Topos Theory.* Adademic Press, London, 1977.

[8] S. MacLane. *Categories for the Working Mathematician*, volume 5. Springer-Verlag, Berlin, 1971.

[9] S.J. Thomas and P.C. Fisher. Nested relational structures. In *Advances in Computing Research, vol. 3, P.C. Kanellakis (editor)*, pages 269–308, 1986.

[10] J.D. Ullman. *Principles of Database and Knowledge-Base Systems.* Computer Science Press, Rockville, Maryland, 1988. vol. I.

[11] G. Vossen. *Data Models, Database Languages and Database Management Systems.* Addison-Wesley, Reading, MA, 1991.