URL: http://www.elsevier.nl/locate/entcs/volume61.html 13 pages

# Sketch Data Models, Relational Schema and Data Specifications ⋆

Michael Johnson

*School of Mathematics and Computing*
*Macquarie University*
*Sydney, Australia*

Robert Rosebrugh

*Department of Mathematics and Computer Science*
*Mount Allison University*
*NB, Canada*

**Abstract**

When different mathematical models are used for software analysis and development it is important to understand their relationships. When the models are truly mathematical, and when the aspects of reality that they seek to model are common, it may be possible to express their relationships in precise mathematical terms. This paper studies three mathematical models: The sketch data model, the relational data model, and the data specifications of Piessens and Steegmans, and determines their relationships mathematically and in detail. The constructions presented here answer reasonably long-standing theoretical questions, and offer techniques that promise to be practically useful in integrating data models.

*Key words:* Category theory, data model, mathematical specification.

## 1 Introduction

This paper is one in a series of papers in which category theoretic methods are used to develop and apply new data models for information system specification, development and research. By way of example, recent results in this work have included a new treatment of the view update problem [10], [17]. The new approach has been applied *inter alia* to database interoperation [16], to the development of enterprise information systems [5], and to software maintenance [18]. In addition, with our colleague Dampney, the techniques have been tested in industry [3], [8],

⋆ Research partially supported by the Australian Research Council, NSERC Canada, and the Oxford Computing Laboratory.

[4]. The category theoretic data model that we use has come to be known as the *sketch data model*.

As the sketch data model has been applied more widely, and particularly with the development of new approaches to problems like the view update problem, practitioners and theoreticians who work with other data models have increasingly been seeking a way of translating between sketch data models and other data models. Translations would allow the sketch data model to be used as one part of a development process involving other data models, and would allow problems (including the view update problem) expressed in other data models to be brought into the framework developed with the sketch data model. Potentially, sketch data model techniques and results might be translated into other models more familiar to practitioners.

The problem addressed by this paper is to make precise the relationship between the sketch data model and other data models. This is essentially a theoretical problem requiring the determination of the mathematical relationships between mathematical models, but it also has important practical applications. At present industrial problems are cast into the sketch data model and analysed there, often without reference to detailed models in other frameworks. A precise connection between for example the relational data model and the sketch data model will allow extant relational models to be translated saving considerable development time.

The paper is organised as follows. In Section 2 we define categories of relational database schema, and of sketch data model schema, and give constructions which (functorially) convert a relational scheme into a sketch scheme and vice versa. We certainly do not expect the constructions to be inverse, or even adjoint, but we do give comparison maps between each of the composites of the two of them and the respective identities. This gives a precise relationship between the schema of the two data models, but we also need to explore the relationship between their models, so in Section 3 we develop an equivalence of categories of certain models (states). Much of the theory of relational databases has been based around normalisation, and in Section 4 we show that the relational states corresponding to sketch data models (instances of sketch data model schema) are in at least third normal form and indicate how a danger to do with transitive dependencies is avoided. This completes our study of relational and sketch data models, and should permit us to relate sketch data models to many others since the relational data model is so ubiquitous.

Having thus dealt with the main question of the paper, the relationship between sketch data models and relational data models, and through them other disparate data models, we move on to consider the relationship between the sketch data model and another model which uses sketches: the *data specifications* of Piessens and Steegmans. In Section 5 we make explicit the relationship between sketch data models and data specifications, and most importantly note that there is not in general an equivalence of model categories for corresponding specifications. We also indicate why the equivalence fails, and we do still obtain a close relationship — an equivalence of one model category with a full subcategory of the other. This

answers another long standing question, that of the precise relationship between these two closely related but heterologous data models.

Finally Section 6 briefly reviews related work, and Section 7 makes some closing remarks.

## 2 Relational schemes and EA sketches

A definition of a relational database scheme includes several components (see, for example, [11]). Here we abstract the main items and describe a category of database schemes.

**Definition 2.1** A *(relational) database scheme* is a triple $S = (D, R, F)$ where

(i) $D = \langle A_d \rangle_{d \in D}$ is a finite family of finite sets — the *domains* of $S$

(ii) $R = \langle 1 \xrightarrow{p_r} C_r \xrightarrow{c_r} D \rangle_{r \in R}$ is a finite family of *primary key - relation schemes*; the elements of the set $C_r$ are the column headings for the table (relation) $r$.

(iii) $F = \langle (s_f, t_f, k_f) \rangle_{f \in F}$ is a finite family of *foreign key definitions*

where $C_r$ is finite for all $r \in R$, $s_f, t_f \in R$, and for all $(s,t,k) \in F$, $1 \xrightarrow{k} C_s$ and $c_s(k) = c_t(p_t)$.

Note that we require a primary key to be a single column heading. The equation at the end of the definition simply requires that the domains of the source and target of a foreign key definition be the same.

**Definition 2.2** A *morphism of database schemes* $G : S \longrightarrow S'$ is specified by mappings $G^D : D \longrightarrow D'$, $G^R : R \longrightarrow R'$, $G^F : F \longrightarrow F'$ and a family of mappings $\langle G(r) : C_r \longrightarrow C'_{r'} \rangle_{r \in R}$ with $r' = G^R(r)$ satisfying

(i) $A^{G_D(d)} = A_d$ (compatibility of domains)

(ii) $G^D c_r = c'_{r'} G(r)$ and $p_{r'} = G(r) p_r$ (compatibility of headings and primary keys)

(iii) denoting $G^F(s,t,k) = (s',t',k')$, $s' = G^R(s)$, $t' = G^R(t)$ and $k' = G(s)k$ (compatibility for foreign keys)

In the sequel, when the component of $G$ is obvious, we will omit the superscript. Evidently, morphisms of database schemes form a category which we denote **DbSch**.

A sketch [2] is a quadruple consisting of a graph, a set of diagrams in the graph, a set of cones in the graph, and a set of cocones in the graph. Let $\Pi$ denote the function taking each cone to its base, and let $\Sigma$ denote the function taking each cocone to its base. We recall from [19] the definition of EA sketch.

**Definition 2.3** An *EA sketch* $\mathbb{E} = (\mathbb{G}, D, L, R)$ is a sketch for which

(i) every $\pi$ in $L$ has $\Pi(\pi)$ finite;

(ii) every $\sigma$ in $R$ has $\Sigma(\sigma)$ finite and discrete;

(iii) there is an object 1 in $\mathbb{G}$ and the cone with empty base and vertex 1 is in $L$.

If $A$ is the vertex of a cocone in $R$, and the objects in the base of that cocone are all 1, then $A$ is called an *attribute*. A node of $\mathbb{G}$ which is neither an attribute nor 1 is called an *entity*. An EA sketch is *keyed* if, for each entity $E$ in $\mathbb{G}$, there is a specified monomorphism $k_E : E \rightarrowtail A_E$, where $A_E$ is an attribute.

All of our EA sketches are henceforth assumed to be keyed and we denote the category of keyed EA sketches and sketch morphisms (that also preserve keys) by **EASk**. We wish to compare **DbSch** and **EASk**, and eventually their model categories.

We begin by noting that it is straightforward to define a functor $S$ from **DbSch** to **EASk**.

**Construction 1:** Let $S = (D, R, F)$ be a database scheme. The sketch $S(S)$ has the following components:

The underlying graph of $S(S)$ has:
(n1) a node denoted $A_d$ for each $d \in D$
(n2) a node denoted $C_r$ for each $r \in R$
(n3) a node denoted 1
and
(e1) an edge from 1 to $A_d$ for each element of the set $A_d$
(e2) an edge from $C_r$ to $A_{c_r(x)}$ for each $x \in C_r$
(e3) an edge from $C_s$ to $C_t$ for each $(s,t,f) \in F$.

The cones $L$ of $S(S)$ are:
(c1) the empty cone with vertex 1
(c2) a monic specification for the edge from $C_r$ to $A_{c_r p_r}$

The discrete cocones $R$ of $S(S)$ are exactly those making the $A_d$ into sums of their elements.

There are no commutative diagrams $D$, and the key for each entity is provided by (c2).

This completes the construction of $S(S)$. □

It is important to notice that we did not add a node to represent the product of the domains determined by $c_r$. The monic specification arising from the primary key for each relation scheme will guarantee that in models of the sketch $S(S)$ the value at any entity is a relation on its attributes.

**Proposition 2.4** *The definition of $S(S)$ extends to morphisms of* **DbSch** *yielding a functor $S$ :* **DbSch** $\longrightarrow$ **EASk**. □

In the other direction and of more interest for our purposes, we obtain a database scheme by ignoring most of the constraint data for an EA sketch.

**Construction 2:** Let $\mathbb{E} = (\mathbb{G}, D, L, R)$ be a keyed EA sketch. The database scheme $T(\mathbb{E})$ has the following data:

The family of domains $D_{\mathbb{E}}$ is indexed by the set $D$ of attribute nodes of $\mathbb{G}$. The domain sets $A_d$ are specified up to isomorphism by the cocones defining the attributes.

The family of relation schemes $R_{\mathbb{E}}$ is indexed by the set $R$ of entities. For each $r \in R$, the set $C_r$ is the set of edges in $\mathbb{G}$ from the node $r$, *except* those edges which are one path of a commutative diagram from $r$ (the exception is discussed in Section 4). The mapping $c_r$ names the attribute nodes with edges from $r$, and the primary key attribute of the target for the edges to entities. The primary key is the selected monic specification.

The family of foreign keys $F_{\mathbb{E}}$ is indexed by the set $F$ of edges in $\mathbb{G}$ among entities (with the exception noted in the previous paragraph). For each $f \in F$, $s_f$ and $t_f$ are the source and target of the edge, and $k_f$ is the element of $C_{s_f}$ associated with the edge.

This completes the construction of $T(\mathbb{E})$. □

**Proposition 2.5** *The definition of $T(\mathbb{E})$ extends to morphisms of* **EASk** *yielding a functor $T$ :* **EASk** $\longrightarrow$ **DbSch**. □

We cannot expect adjunction between $S$ and $T$, but their composites are related to the identity functors.

First, note that we can easily define a comparison morphism

$$\varphi_S : TS(S) \longrightarrow S$$

in **DbSch**. Both $T$ and $S$ preserve attribute data, so the domain information for $TS(S)$ is exactly that of $S$. Construction 1 defines an entity of $S(S)$ for each relation scheme of $S$, and Construction 2 defines a relation scheme for each entity of $\mathbb{G}$. Thus, the set of relation schemes for $TS(S)$ is the same as that for $S$. Primary keys are preserved. However, Construction 2 introduces a new column head in $C_r$ whenever there is a foreign key from a relation scheme. The morphism $\varphi_S$ should identify new headings which so result. Finally, the foreign keys of $TS(S)$ arise exactly from those of $S$. Note that because there are no commutative diagrams in $S(S)$, the exception to edges giving foreign keys in Construction 2 does not apply to $S(S)$.

On the other hand, we have a comparison morphism of EA sketches

$$\psi_{\mathbb{E}} : ST(\mathbb{E}) \longrightarrow \mathbb{E}.$$

As just noted, the attribute and primary key data is preserved by both constructions. Indeed, by Constructions 1 and 2 the underlying graph of $ST(\mathbb{E})$ has exactly the same nodes $\mathbb{G}$. It has an edge for each edge of $\mathbb{G}$ except those to which the exception in Construction 2 applies. $ST(\mathbb{E})$ has none of the commutative diagrams of $\mathbb{E}$. Its finite limit constraints arise only from monic specifications appearing as keys in $\mathbb{E}$, and its cocones are exactly those defining attributes.

## 3 Models

Our interest in this paper is solely in models with values in the category of finite sets **set$_0$**. We denote the category of **set$_0$** models of an EA sketch by $\mathrm{Mod}(\mathbb{E})$ and in this article do not call them database states in order to distinguish them from objects of $\mathrm{St}(S)$ as defined below. Recall from [19] that for a keyed EA sketch $\mathbb{E}$, $\mathrm{Mod}(\mathbb{E})$ is a preorder and all components of a morphism of models are monic.

For a database scheme $S = (D, R, F)$ we need to define the category of states of $S$, which we will denote $\mathrm{St}(S)$. It too will be a preorder.

**Definition 3.1** A *state of* $S$ is a family $D = \langle D_r \rangle_{r \in R}$ of relations indexed by $R$ satisfying, for $r \in R$ and $f = (s,t,k) \in F$:
(s1) $D_r$ has signature $C_r$, i. e. $D_r$ is a subset of $\Pi_{x \in C_r} A_{c_r(x)}$
(s2) (primary key integrity) $p_r$ is a key, i. e. the following composite is monic:

$$D_r \hookrightarrow \Pi_{x \in C_r} A_{c_r(x)} \xrightarrow{\pi} A_{c_r p_r}$$

(s3) (foreign key integrity) the projection $D_k^s$ of $D_s$ on $A_k = A_{p_t}$ is included in the projection of $D_t$ on $A_{p_t}$ as below (where $D_s^k$ and $D_t^{p_t}$ are images of the composite of the other two mappings in their square)

$$\begin{array}{ccccccc}
D_s & \longrightarrow & D_s^k & - - \rightarrow & D_t^{p_t} & \longleftarrow & D_t \\
\downarrow & & \downarrow & & \downarrow & & \downarrow \\
\Pi_{i \in C_s} A_{c_s(i)} & \longrightarrow & A_{c_s(k)} & \xrightarrow{=} & A_{c_t p_t} & \longleftarrow & \Pi_{j \in C_t} A_{c_t(j)}
\end{array}$$

**Definition 3.2** A *morphism of states* $D \xrightarrow{m} D'$ of $S$ is a family $\langle D_r \xrightarrow{m_r} D'_r \rangle_{r \in R}$ of functions which commute with the inclusions of $D_r$ and $D'_r$ in $\Pi_{i \in C_r} A_i$.

Note that the $m_r$ are necessarily monic. The states and morphisms of $S$ determine the category $\mathrm{St}(S)$. Given a model $M$ of an EA sketch $\mathbb{E}$ with non-attribute entities $\langle E_r \rangle_{r \in R}$, it is clear how to view the family $\langle M(E_r) \rangle_{r \in R}$ as a family of relations that is a state of $T(\mathbb{E})$. Similarly, morphisms of models of $\mathbb{E}$ determine morphisms in $\mathrm{St}(T(\mathbb{E}))$, and we obtain:

**Proposition 3.3** *If* $\mathbb{E}$ *is an EA sketch, there is a fully faithful functor*

$$C : \mathrm{Mod}(\mathbb{E}) \longrightarrow \mathrm{St}(T(\mathbb{E}))$$

Indeed, since $\mathbb{E}$ is keyed, any morphism of states from $C(M)$ to $C(N)$, say arises from a morphism of models. There is also a functor

$$\psi_{\mathbb{E}}^* : \mathrm{Mod}(\mathbb{E}) \longrightarrow \mathrm{Mod}(ST(\mathbb{E}))$$

induced by the sketch morphism $\psi_{\mathbb{E}} : ST(\mathbb{E}) \longrightarrow \mathbb{E}$ noted in the previous section.

**Proposition 3.4** *There is an equivalence of categories*

$$J : \mathrm{St}(T(\mathbb{E})) \longrightarrow \mathrm{Mod}(ST(\mathbb{E}))$$

*satisfying* $\psi^*_{\mathbb{E}} = JC$.

**Proof.** As noted at the end of Section 2, $ST(\mathbb{E})$ has the attribute and primary key data of the database scheme $T(\mathbb{E})$. To define $J$, note that a family of relations which is a state $D$ of $T(\mathbb{E})$ allows taking the domains of the relations as values of a model $M_D$ of $ST(\mathbb{E})$ at the entities. Foreign key satisfaction defines the model $M_D$ on edges of the sketch $ST(\mathbb{E})$, and there are no commutativities or additional limit constraints to check. In the other direction, for a model $M$ of $ST(\mathbb{E})$, the set $M(E)$ at an entity $E$ is certainly a relation on the domains of the relation scheme $C_E$ in $T(\mathbb{E})$ because of the key constraint. Moreover, a model will satisfy all of the foreign key constraints of $T(\mathbb{E})$ Thus we get a database state $D_M$ in $\mathrm{St}(T(\mathbb{E}))$. On both sides of the equivalence, homomorphisms are defined by compatible families of monos and the categories are preorders. We leave verification that the constructions outlined are essentially mutually inverse to the reader. □

## 4 Normalisation

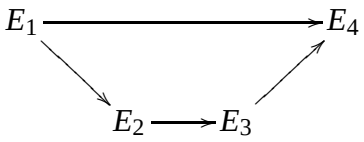
In principle the database scheme arising from a keyed EA sketch should be close to being normalised. Recall that functional dependencies are specified from one set of attributes to another. The edges between nodes in the underlying graph of a sketch express functional dependency between their key attributes. We have sometimes insisted that attributes not be the domains of arrows. In that case the EA sketch *does not express any functional dependency except:*

(i) those from the key attribute of the source of an edge to the key attribute of its target

(ii) the dependency of a non key attribute on a primary key.

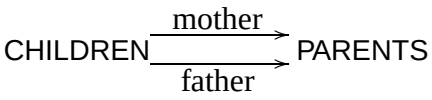
Thus, if there are other functional dependencies an EA sketch *cannot express* them. This appears to be a weakness of the model since we cannot expect the absence of such functional dependencies. However, if we assume that all functional dependencies *are* expressed by the EA sketch $\mathbb{E}$, then the relation schemes of $T(\mathbb{E})$ will be in 3NF (indeed BCNF if there are no monic constraints from entities other than specified keys).

The reason for the exception in Construction 2 is to avoid having a transitive functional dependency arise in $T(\mathbb{E})$. When there is a commuting diagram consisting of a single edge and another commuting path from an entity $E_1$ a transitive dependency arises. For example suppose the following commutes in $\mathbb{E}$




Here, if $A_2$ and $A_4$ are the key attributes of $E_2$ and $E_4$, we have that $A_2 \longrightarrow A_4$ is a derived dependency which would be transitive had we added the single edge from $E_1$ to $E_4$ to $C_{E_1}$.

The above situation appears to be the only one causing potential problems. Note that if there are two (or more) edges from $E_1$ to a second entity $E_2$, they simply create two attributes and two foreign keys, e. g.




Moreover, if two paths from $E_1$ with the same target both have length greater than one, the problem does not arise. Indeed, the target entity does not contribute an attribute to $C_{E_1}$

The considerations above justify:

**Proposition 4.1** *With the dependencies implied by an EA sketch $\mathbb{E}$, the database scheme $T(\mathbb{E})$ is 3NF.*

## 5 EA sketches and data specifications

In their articles [21,22], Piessens and Steegmans have proposed the notion of *data specification* for semantic data modelling. Here we relate models for this view of data modelling with our EA sketch data models. We modify Piessens' definition to require that attributes have finite values since we are unconvinced by his arguments for so-called 'meta-finite' structures in Computer Science.

**Definition 5.1** A *data specification* $(S,A)$ consists of

(i) a finite sketch $S = (G,\mathbf{D},L,C)$

(ii) for each $T$ in $V$, the nodes of $G$, a finite set $A_T$, called the *set of attributes* of entity type $T$, and for each $a \in A_T$ an *attribute set* $S_T(a)$, and we denote these collectively by $A(T) = \Pi_{a\in A_T} S_T(a)$.

So $A$ can be viewed as a functor $A : V \longrightarrow \mathbf{set}$ (V the discrete category), which is used in the following definitions. Denote the obvious inclusions $I : V \longrightarrow G$ ($V$ the discrete graph) and $J : \mathbf{set}_0 \longrightarrow \mathbf{set}$.

**Definition 5.2** A *model* of a data specification $(S,A)$ is a couple $(M,\lambda)$ where $M$ is a model of $S$ in $\mathbf{set}_0$ and $\lambda : JMI \longrightarrow A$ is a natural transformation.

**Definition 5.3** A *homomorphism* from a data specification model $(M,\lambda)$ to a data specification model $(M',\lambda')$ is a natural transformation $\phi : M \longrightarrow M'$ satisfying the equation $\lambda' \circ I\phi J = \lambda$.

In order to compare EA sketches and data specifications we first note that the latter permit non-discrete cocones. We do not include such cocones in EA sketches since they would, for example, permit constraints requiring identification of values of base entities.

**Proposition 5.4** *Let $(S,A)$ be a data specification whose cocones $C$ are discrete. Then there is an EA sketch $\mathbb{E}$ such that* Mod$(\mathbb{E})$ *is equivalent to the category of models of $(S,A)$.*

**Proof.** Let $(S,A)$ be a data specification as in the statement. First augment its sketch by adding an attribute cocone for each attribute set $S_T(a)$ in the specification and an arrow from $T$ to the vertex of the cocone. The resulting sketch $\mathbb{E}_{(S,A)}$ is clearly an EA sketch.

Now any model $M$ of $\mathbb{E}_{(S,A)}$ clearly defines a model of the data specification $(S,A)$: restrict $M$ to $S$ and define $\lambda$ using the $M$ components of the arrows to $S_T(a)$ attributes. Any morphism of states defines a homomorphism of models. Moreover, any model $(M,\lambda)$ of $(S,A)$ is the image under the correspondence just defined of a state of $\mathbb{E}_{(S,A)}$. The state has the values of $M$ on $S$ and $\lambda$ serves to define the required arrows from $M(T)$ to the (vertices of cocones constructed from the) $S_T(a)$. Thus we have defined a functor from $\mathrm{Mod}(\mathbb{E}_{(S,A)})$ to models of $(S,A)$ which is surjective on objects. To complete the demonstration of the claimed equivalence requires only noting that the correspondence is fully faithful. □

A slightly subtle point we should make explicit above is that the $\lambda_T$ defined from a state are more precisely (but ignoring Piessens' $I$ and $J$)

$$M(T) \longrightarrow \Pi M(S_T(a)) \cong A(T)$$

and in saying that we have a functor we are assuming that the isos are chosen coherently. Note that we obtain an equivalence rather than an isomorphism here.

**Proposition 5.5** *Let $\mathbb{E}$ be an EA sketch such that*

(i) *no attribute is the codomain of a commutative diagram*

(ii) *no attribute is in the base of a cone.*

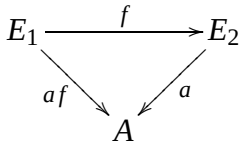
*There is data specification $(S_{\mathbb{E}},A_{\mathbb{E}})$ whose category of models is isomorphic to* $\mathrm{Mod}(\mathbb{E})$.

**Proof.** Let $\mathbb{E}$ be an ER sketch as in the statement. Modify $\mathbb{E}$ to $S_{\mathbb{E}}$ by removing attributes, their defining cocones and any arrows to them. Call the resulting sketch $S_{\mathbb{E}}$. Then, for each of the remaining entities $T$, and each arrow $a$ from $T$ to an attribute in $\mathbb{E}$, add $a$ to $A_T$ and define $S_T(a)$ to be a set with as many elements as the attribute at the codomain of $a$. This defines $(S_{\mathbb{E}},A_{\mathbb{E}})$.

Now a model $M$ for $\mathbb{E}$ in $\mathbf{set}_0$ has finite values and satisfies the constraints of $S_{\mathbb{E}}$, so that it restricts to a unique model $M_{\mathbb{E}}$ of $S_{\mathbb{E}}$ in $\mathbf{set}_0$. The values of $M$ on attributes are determined to within isomorphism and consistent with $A_{\mathbb{E}}$, so the $M(a) : M(T) \longrightarrow S_T(a)$ for $a \in A_T$ combine to define $\lambda_T : M_{\mathbb{E}}(T) \longrightarrow A(T)$. There are no naturality conditions to check, so $M_{\mathbb{E}}$ and the $\lambda$'s determine a unique model of $(S_{\mathbb{E}},A_{\mathbb{E}})$. Moreover any homomorphism of models of $\mathbb{E}$ determines a unique homomorphism of models of $(S_{\mathbb{E}},A_{\mathbb{E}})$. Conversely, any model $(M,\lambda)$ of $(S_{\mathbb{E}},A_{\mathbb{E}})$ can be extended to a model $M'$ in $\mathrm{Mod}(\mathbb{E})$ since the assumptions (i) and (ii) guarantee that the $\lambda_T$ can be used to define $M'$ on the arrows deleted from $\mathbb{E}$ in the passage to the data specification and a homomorphism $M_1 \longrightarrow M_2$ of models can be extended to a unique morphism of models of $\mathbb{E}$. □

The need for conditions (i) and (ii) results from what we see as a gap in the notion of data specification. Suppose that $E_1$ and $E_2$ are entities and $A$ is an attribute

of $\mathbb{E}$. A very natural commutativity for $\mathbb{E}$ to include is a triangle:




A data specification model can impose this requirement only by ignoring $af$. Worse still, if $A$ is the codomain of a commutative square or pullback, say, with attributes at its corners, there is simply no way to allow $A$ to be an attribute of the corners in a data specification and enforce the constraints. Thus we can only obtain the following by weakening these conditions and using the construction above.

**Proposition 5.6** *Let $\mathbb{E}$ be an EA sketch. There is a data specification $(S_{\mathbb{E}}, A_{\mathbb{E}})$ whose category of models has a full subcategory isomorphic to* $\mathrm{Mod}(\mathbb{E})$.

## 6 Related Research

In the last decade there has been considerable growth in the use of category theory to support semantic data modelling. Piessens and Steegmans used data specifications as described above to obtain results on the algorithmic determination of equivalences of model categories [21] [22] which were intended to support plans for view integration. Diskin and Cadish have used sketches for a variety of modelling purposes including for example [12] and [13]. They have been concentrating on developing the diagrammatic language of "diagram operations". Several others, including Lippe and ter Hofstede [20], Islam and Phoa [14], Tuijn and Gyssens [24], Rosebrugh and Wood [23] and Baclawski et al [1], have been using category theory for data modelling.

While some of these authors have attempted to use category theory to study particular data models, including the relational data model, none has given a precise statement of the relational data model in category theoretic terms, and to the authors' knowledge there is no previous work aimed at obtaining explicit translations between ordinary and category theoretic data models. (Probably this is not surprising, since the need for such translations only becomes apparent once the category theoretic models have been applied in industry sufficiently widely.)

In parallel with this work, the authors and others have been exploring further aspects of the sketch data model. Johnson, Rosebrugh and Wood have discovered a mathematical foundation using *equipments* that unifies the treatment of the categories of database states, the query language (as treated in [9]), and updates, as instances of modelling sketches in 2-categories [19]; Dampney and Johnson have developed a database interoperation technique that reduces the need for attribute harmonisation across interoperating data models [6]; and in less mathematical work they have proposed the category theoretic data modelling techniques as a foundational ontology for information systems research [7].

## 7 Conclusion

The propositions presented above answer the questions that motivated this paper. It remains to remark on a couple of points that could lead to confusion.

First, we have compared a model based on category theory, the sketch data model, with another model, the relational model, by presenting the schema and models of the latter as categories. This is not to say that we have converted the relational model into a category theoretic model for our purposes. Rather, we have used the technique common in mathematics of studying the relationships between mathematical structures by studying the categories of those structures (for example, algebraic topology studies categories of topological spaces, categories of algebras, and functors between them). It is merely a happenstance that one of the mathematical structures, the sketch data model, was already based upon category theory.

Secondly, the term "model" is remarkably overloaded in these fields. Rather than introducing new terminology to try to distinguish the different uses, we merely catalogue them here and leave it to context to disambiguate them. At the highest level, the act of searching for a mathematical representation of things in the world is called *modelling*. When the representations will be specifications of datatypes, and when the datatypes can be chosen from a certain collection as part of a particular modelling methodology, the collection and its corresponding methodology are called *data models*. Thus the sketch data model, and the relational data model depend upon certain basic datatypes (EA sketches and relations respectively). Frequently the result of a modelling exercise is referred to as *a* data model. For example a relational schema is sometimes called a data model of the particular real-world domain being studied. Finally a particular item, a snap-shot of a database for example, which is an instance of a data model, is often called a model. This last usage agrees with the use of model in logic and other fields — a model of a theory.

To summarise with an example "These data are a model of a data model that I developed using the relational data model when I wanted to model the information required to . . . "

## References

[1] K. Baclawski, D. Simovici and W. White. A categorical approach to database semantics. *Mathematical Structures in Computer Science*, 4:147–183, 1994.

[2] M. Barr and C. Wells. *Category theory for computing science*. Prentice-Hall, second edition, 1995.

[3] C. N. G. Dampney and Michael Johnson. TIME Compliant Corporate Data Model Validation. Consultants' report to Telecom Australia, 1991.

[4] C. N. G. Dampney and Michael Johnson. Fibrations and the DoH Data Model. Consultants' report to NSW Department of Health, 1999.

[5] C. N. G. Dampney and Michael Johnson. Enterprise Information Systems: Specifying the links among project data models using category theory. Proceedings of the International Conference on Enterprise Information Systems, 619–626, 2001.

[6] C. N. G. Dampney and Michael Johnson. Half-duplex interoperations for cooperating information systems. In *Advances in Concurrent Engineering*, IN-1, 7pp, International Institute of Concurrent Engineering, ISBN 09710461-0-7, 2001.

[7] C. N. G. Dampney and Michael Johnson. On Category Theory as a (meta)-Ontology for Information Systems Research. In press for Formal Ontology in Information Systems (FOIS), Maine, October 2001, ACM Publications.

[8] C. N. G. Dampney, Michael Johnson and G. M. McGrath. Audit and Enhancement of the Caltex Information Strategy Planning (CISP) Project. Consultants' report to Caltex Oil Australia, 1993.

[9] C. N. G. Dampney, Michael Johnson, and G. P. Monro. An illustrated mathematical foundation for ERA. In *The unified computation laboratory*, pages 77–84, Oxford University Press, 1992.

[10] C. N. G. Dampney, Michael Johnson, and Robert Rosebrugh. View Updates in a Semantic Data Model Paradigm. ADC2001, 29–36, IEEE Press, 2001.

[11] C. J. Date. *Introduction to Database Systems, Sixth Edition*. Addison-Wesley, 1995.

[12] Zinovy Diskin and Boris Cadish. Algebraic graph-based approach to management of multidatabase systems. In *Proceedings of The Second International Workshop on Next Generation Information Technologies and Systems (NGITS '95)*, 1995.

[13] Zinovy Diskin and Boris Cadish. Variable set semantics for generalised sketches: Why ER is more object oriented than OO. In *Data and Knowledge Engineering*, 2000.

[14] A. Islam and W. Phoa. Categorical models of relational databases I: Fibrational formulation, schema integration. Proceedings of the TACS94. Eds M. Hagiya and J. C. Mitchell. *Lecture Notes in Computer Science*, 789:618–641, 1994.

[15] Michael Johnson and C. N. G. Dampney. On the value of commutative diagrams in information modelling. In The Unified Computation Laboratory, eds Rattray and Clarke, *Springer Workshops in Computing*, 77–84, Springer-Verlag, 1994.

[16] Michael Johnson and Robert Rosebrugh. Database interoperability through state based logical data independence. *International Journal of Computer Applications in Technology*, in press, 2001.

[17] Michael Johnson and Robert Rosebrugh. View updatability based on the models of a formal specification. *Proceedings of Formal Methods Europe 2001*, 534-549, *Lecture Notes in Computer Science* 2021, 2001.

[18] Michael Johnson and Robert Rosebrugh. Reverse Engineering Legacy Information Systems for Internet Based Interoperation. In press for the International Conference on Software Maintenance, Florence, November, 2001.

[19] Michael Johnson, Robert Rosebrugh and R. J. Wood. Entity-relationship-attribute models and sketches. Submitted to *Theory and Applications of Categories*.

[20] E. Lippe and A ter Hofstede. A category theoretical approach to conceptual data modelling. *RAIRO Theoretical Informatics and Applications*, 30:31–79, 1996.

[21] Frank Piessens and Eric Steegmans. Categorical data specifications. *Theory and Applications of Categories*, 1:156–173, 1995.

[22] Frank Piessens and Eric Steegmans. Selective Attribute Elimination for Categorical Data Specifications Proceedings of the 6th International AMAST. Ed. Michael Johnson *Lecture Notes in Computer Science*, 1349:424-436, 1997.

[23] Robert Rosebrugh and R. J. Wood. Relational databases and indexed categories. In *CMS Conference Proceedings*, 13:391–407, American Mathematical Society, 1992.

[24] C. Tuijn and M. Gyssens. CGOOD, a categorical graph-oriented object data model. *Theoretical Computer Science*, 160:217-239, 1996.