Theory and Applications of Categories, Vol. 1, No. 8, 1995, pp. 156–173.

CATEGORICAL DATA-SPECIFICATIONS

FRANK PIESSENS AND ERIC STEEGMANS
Transmitted by Jiri Rosicky

ABSTRACT. We introduce MD-sketches, which are a particular kind of Finite Sum sketches. Two interesting results about MD-sketches are proved. First, we show that, given two MD-sketches, it is algorithmically decidable whether their model categories are equivalent. Next we show that data-specifications, as used in database-design and software engineering, can be translated to MD-sketches. As a corollary, we obtain that equivalence of data-specifications is decidable.

1. Introduction

This paper is about the correspondence between two kinds of specification mechanisms: sketches and data-specifications.

Sketches were invented by Charles Ehresmann in the late sixties, for the purpose of specifying algebraic structures. Since then, they have been studied intensively. An outline of the theory of sketches, with a large number of references, can be found in [17]. It is well-known that the categories of models of sketches (in **Set**) are exactly the accessible categories. An accessible category is complete iff it is the model category of a Limit sketch iff it is a locally presentable category. Locally presentable categories have the pleasant property that it is possible to define a canonical Limit sketch with the given category as model category. All these results can be found in [2]. For other accessible categories, it is not possible to define a canonical sketch with the given category as model category. In a number of recent papers ([3, 4]), various equivalence preserving transformations between sketches are studied. These transformations preserve (up to equivalence) the model category in **Set**, but do not necessarily preserve model categories in other categories than **Set**. Hence, it is clear that model categories in **Set** do not determine the sketch in the same way as this is the case for Limit sketches. In this paper we are interested in algorithmic decidability of the equivalence problem, and we will prove that the equivalence problem is decidable for the class of MD-sketches.

Semantic data-specifications like Chen's Entity-Relationship diagrams ([10]) on the other hand, have been used for many years in the early stages of database design. However, most specification systems used in practice are of an informal nature, and have little or no mathematical foundation. This makes it impossible to prove interesting results about them. [13] formalized these specification systems using categorical language, and proved

The first author is a Research Assistant of the Belgian Fund for Scientific Research
Received by the editors 19 June 1995.
Published on 2 November 1995
1991 Mathematics Subject Classification : 18A25, 18C99, 68P15.
Key words and phrases: data-specifications, sketches.

a first interesting result: the existence of so-called *canonical forms* for a large subset of data-specifications. Other categorical formalisations of semantic data-specifications exist. [9] and [11] propose generalized sketches in the sense of [12] as a formalisation for data-specifications. In [15], object-oriented data-specification systems based on categorical constructs are defined, and a query language based on universal constructions is shown to be at least as expressive as relational algebra.

The structure of this paper is as follows:

First, in section 2, we introduce a special kind of Finite Sum sketch ([5]), in which you can only specify certain monicity and disjointness constraints. This kind of sketch is called an MD-sketch. The main result about MD-sketches is that equivalence of the model-categories of two MD-sketches is decidable.

In section 3 we define data-specifications, and their model-categories. Our definition is equivalent to the definition of *specification with attributes* in [13]. If the model-categories of two different data-specifications are equivalent, this means that they are different formalizations of the same real world situation. Recognizing that two different data-specifications are equivalent is very important when one tries to integrate a number of data-specifications into one large data-specification. Hence, it would be interesting to have an algorithm to decide equivalence of data-specifications.

In section 4, we show that data-specifications can be translated to MD-sketches, and as a corollary, we obtain that equivalence of data-specifications is decidable. Combining the translation algorithm with the algorithm to decide equivalence of MD-sketches gives us an algorithm to decide equivalence of data-specifications.

1.1. Acknowledgments. We want to thank Francis Borceux for a number of very clarifying discussions about accessible categories and about Kan-extensions, and Jiří Rosický for a very interesting discussion about the equivalence problem for sketches when one considers models in **Set** or in **FinSet**. We also want to thank Dominic Verity, Roy Crole and all the other readers of the categories mailing list who took the time and effort to answer our questions about discrete opfibrations.

2. MD-sketches

An MD-sketch is a peculiar kind of Finite Sum sketch. MD-sketches are interesting because equivalence of their model categories is algorithmically decidable (theorem 2.18), and yet they have sufficient expressive power to allow for a translation of data-specifications to MD-sketches (theorem 4.3).

2.1. Preliminary Definitions.

2.2. Definition. *A (finite)* source *in a category $\mathcal{C}$ is a pair $(X,(f_i)_{i\in I})$ consisting of an object X of $\mathcal{C}$ and a family of morphisms $f_i : X \to Y_i$ of $\mathcal{C}$, indexed by some (finite) set I.*

In this paper, we only consider finite sources, and we will take the unqualified word "source" to mean *finite* source. We will use the notations $(X,(f_i)_{i\in I})$ and $f_i : X \to Y_i$ interchangeably. A source $(X,(f_i)_{i\in I})$ is a *mono-source* if $f_i \circ x = f_i \circ y$ for all $i \in I$

implies that $x = y$. The *base* of a source $f_i : X \to Y_i$ is the indexed family of objects Y_i. Hence, the base of a source is a discrete diagram.

These definitions of source and mono-source in a category are standard, and can be found for instance in [1].

2.3. Definition. *A* double-source *in a category $\mathcal{C}$ is a pair of two sources $f_i : X \to Y_i$ and $g_i : Z \to Y_i$ in $\mathcal{C}$, on the same base.*

A double-source (f_i, g_i) looks like this:

$$\begin{tikzcd}
 & Y_1 & \\
X \arrow[ur, "f_1"] \arrow[dr, "f_n"'] & \vdots & Z \arrow[ul, "g_1"'] \arrow[dl, "g_n"] \\
 & Y_n &
\end{tikzcd}$$

We say that a double-source (f_i, g_i) is *disjoint*, iff the limit of the diagram above is the initial object. In **FinSet**, the category of finite sets and functions, this means that there are no elements $x \in X$ and $z \in Z$ such that $f_i(x) = g_i(z)$ for all i.

2.4. Definition. *An* MD-sketch *is a triple $(\mathcal{C}, \mathcal{M}, \mathcal{D})$ where $\mathcal{C}$ is a finite category, $\mathcal{M}$ is a finite set of sources in $\mathcal{C}$, $\mathcal{D}$ is a finite set of double-sources in $\mathcal{C}$.*

2.5. Definition. *A* model *of an MD-sketch $(\mathcal{C}, \mathcal{M}, \mathcal{D})$ is a functor from $\mathcal{C}$ to* **FinSet**, *which takes every source in $\mathcal{M}$ to a mono-source and every double-source in $\mathcal{D}$ to a disjoint double-source.*

A source μ will sometimes be called a monicity condition. A monicity condition μ is satisfied by a functor F iff F takes μ to a mono-source. In a similar way, a double-source δ will sometimes be called a disjointness condition, and a disjointness condition δ is satisfied by a functor F iff F takes δ to a disjoint double-source.

The *model-category* of an MD-sketch, denoted $\mathrm{Mod}(\mathcal{C}, \mathcal{M}, \mathcal{D})$ is the full subcategory of $\mathrm{Fun}(\mathcal{C}, \mathbf{FinSet})$ containing only the models. We use the notation $\mathrm{Mod}(\mathcal{C}, \mathcal{M})$ to denote the full subcategory of $\mathrm{Fun}(\mathcal{C}, \mathbf{FinSet})$ consisting of all those functors that take all sources in $\mathcal{M}$ to mono-sources (but do not necessarily satisfy the disjointness conditions).

2.6. Remark. It should be obvious that an MD-sketch is a restricted kind of Finite Sum sketch ([5]). The requirement that a source must be mono can be stated by requiring a certain cone to be a limit cone. To state the disjointness conditions, you add a new object to $\mathcal{C}$, and an arrow from the new object to every object of $\mathcal{C}$. Then a disjointness condition can be stated by requiring the new object to be initial, and by requiring that the limit of the double-source is this new object.

We could consider models of MD-sketches in other categories than **FinSet**. However, for the application that we have in mind (data-specifications), only models in **FinSet** are needed. Therefore, we restrict our attention to models in **FinSet**.

Example. We give an example to show that model categories of MD-sketches are not necessarily finitely complete. Since model categories of Finite Limit sketches are always finitely complete, it follows that an MD-sketch is not a restricted kind of Finite Limit sketch. Consider the following MD-sketch:

$$\mathcal{C} = \begin{tikzcd} X \arrow[dr, "f"'] & & Y \arrow[dl, "g"] \\ & Z & \end{tikzcd}, \quad \mathcal{M} = \emptyset, \quad \mathcal{D} = \{(f,g)\}$$

(Only the non-identity arrows of the category $\mathcal{C}$ are drawn). Models of this MD-sketch are functors $F : \mathcal{C} \to \mathbf{FinSet}$ such that the images of Ff and Fg are disjoint. It is easy to see that the category of models does not have a terminal object.

2.7. Properties of MD-Sketches.

2.8. Lemma. $\mathrm{Mod}(\mathcal{C}, \mathcal{M})$ *is an epi-reflective subcategory of* $\mathrm{Fun}(\mathcal{C}, \mathbf{FinSet})$. *Moreover, there exists an algorithm to compute the reflection of a functor* $F : \mathcal{C} \to \mathbf{FinSet}$.

The algorithm to compute the reflection is given in the proof below.

Proof. The construction of the reflection of a functor F is as follows:

1. Define a relation R on the elements of F where xRy iff there exists some source $f_i : X \to Y_i \in \mathcal{M}$ such that $Ff_i(x) = Ff_i(y)$ for all i.

2. Construct the smallest congruence relation on the elements of F, containing R.

3. Take the quotient of F by this congruence relation.

The universal arrow from F to its reflection is the projection of F on this quotient. We leave the verification to the reader. ∎

Hence, we have the following situation:

$$\mathrm{Fun}(\mathcal{C}, \mathbf{FinSet}) \underset{I}{\overset{R}{\rightleftarrows}} \mathrm{Mod}(\mathcal{C}, \mathcal{M}) \xleftarrow[J]{} \mathrm{Mod}(\mathcal{C}, \mathcal{M}, \mathcal{D}) \tag{1}$$

where I and J are inclusions and R is left adjoint to I.

2.9. Lemma. *Suppose F is in the range of J (i.e. $F = J(G)$ for some G), and suppose there exists a morphism $\alpha : F' \to F$ in* $\mathrm{Mod}(\mathcal{C}, \mathcal{M})$, *then F' is also in the range of J.*

Proof. Suppose F' is not in the range of J. This means that there exists a disjointness condition $(f_i : X \to Y_i, g_i : Z \to Y_i)$ in $\mathcal{D}$ which is not satisfied by F'. Hence, there exists elements $x' \in F'(X)$ and $z' \in F'(Z)$ such that $F'(f_i)(x') = F'(g_i)(z')$ for all i. But then, let $x = \alpha_X(x')$ and $z = \alpha_Z(z')$, and we find that $F(f_i)(x) = F(g_i)(z)$ for all i. As a consequence, the disjointness condition (f_i, g_i) is not satisfied by F and F cannot be in the range of J. Contradiction. ∎

2.10. Lemma. *J preserves and reflects colimits.*

Proof. Since J is full and faithfull, it is obvious that it reflects colimits. To prove that it preserves colimits, consider a diagram D in $\mathrm{Mod}(\mathcal{C}, \mathcal{M}, \mathcal{D})$, and suppose it has a colimit (L, ι). We must prove that the colimit of $J(D)$ is $(J(L), J(\iota))$. Since $\mathrm{Mod}(\mathcal{C}, \mathcal{M})$ is cocomplete, $J(D)$ must have a colimit (L', ι'). By the universal property of the colimit, we find an arrow $\alpha : L' \to J(L)$ such that $\alpha \circ \iota' = J(\iota)$. But, by the previous lemma, this means that L' is in the range of J. Since J reflects colimits, we conclude that L' and $J(L)$ are isomorphic. ∎

2.11. Definition. *An MD-sketch $(\mathcal{C}, \mathcal{M}, \mathcal{D})$ is* normal*, or is in* normal form*, if all the representable functors from $\mathcal{C}$ are models.*

2.12. Lemma. *For every MD-sketch, there exists a normal MD-sketch with an equivalent model category. Moreover, there exists an algorithm to compute such a normal MD-sketch.*

An algorithm to compute an equivalent normal MD-sketch is contained in the proof below.

Proof. Suppose an MD-sketch $(\mathcal{C}, \mathcal{M}, \mathcal{D})$ is given. Consider diagram (1), and let η be the unit of the adjunction (R, I). We define a congruence relation on the arrows of $\mathcal{C}$ in the following way:

$$f \sim g \quad \text{iff} \quad \eta_{\mathrm{Hom}(C,-)}(f) = \eta_{\mathrm{Hom}(C,-)}(g)$$

with C the source of f and g. Let $\mathcal{C}'$ be $\mathcal{C}/\sim$ and let $P : \mathcal{C} \to \mathcal{C}/\sim$ be the projection. Since $\mathrm{Nat}(\mathrm{Hom}(C, -), IM) = \mathrm{Nat}(IR\mathrm{Hom}(C, -), IM)$, we can conclude that the arrows that are identified by $\sim$ must be taken to the same function in every model. Define $\mathcal{M}'$ to be $\{P(\mu) \mid \mu \in \mathcal{M}\}$ and $\mathcal{D}'$ to be $\{P(\delta) \mid \delta \in \mathcal{D}\}$. It is easy to verify that $\mathrm{Mod}(\mathcal{C}, \mathcal{M}, \mathcal{D})$ is equivalent to $\mathrm{Mod}(\mathcal{C}', \mathcal{M}', \mathcal{D}')$, and that all representable functors from $\mathcal{C}'$ satisfy all the monicity conditions in $\mathcal{M}'$.

Next, suppose that $\mathrm{Hom}(C', -)$ does not satisfy a disjointness condition δ in $\mathcal{D}'$. By lemma 2.9, it follows that all models of $(\mathcal{C}', \mathcal{M}', \mathcal{D}')$ must take C' to the empty set. Hence, $\mathrm{Mod}(\mathcal{C}', \mathcal{M}', \mathcal{D}')$ is equivalent to $\mathrm{Mod}(\mathcal{C}'', \mathcal{M}'', \mathcal{D}'')$ with:

- $\mathcal{C}''$ is the full subcategory of $\mathcal{C}'$ containing all the objects X for which $\mathrm{Hom}(X, -)$ is a model of $(\mathcal{C}', \mathcal{M}', \mathcal{D}')$.
- $\mathcal{M}'' = \{f_i : X \to Y_i \in \mathcal{M}' \mid X \in \mathcal{C}''\}$
- $\mathcal{D}'' = \{(f_i : X \to Y_i, g_i : Z \to Y_i) \in \mathcal{D}' \mid X \in \mathcal{C}'' \text{ and } Z \in \mathcal{C}''\}$

It is easy to verify that $(\mathcal{C}'', \mathcal{M}'', \mathcal{D}'')$ is normal, and that its model category is equivalent to that of $(\mathcal{C}', \mathcal{M}', \mathcal{D}')$, and hence to that of $(\mathcal{C}, \mathcal{M}, \mathcal{D})$.

Since $\mathcal{C}$ is a finite category, all these constructions are clearly computable. ∎

Example. Consider the following MD-sketch:

$$\mathcal{C} = \begin{tikzcd} B \arrow[dr, "f"] & & \\ & C \arrow[r, "i"] & D \\ A \arrow[ur, "g"] \arrow[ur, shift right, "h"'] & & \end{tikzcd}, \quad \mathcal{M} = \{i\}, \quad \mathcal{D} = \{(f, f)\}$$

with $i \circ h = i \circ g$. It is clear that $\mathrm{Hom}(A, -)$ does not satisfy the monicity conditions, since $\mathrm{Hom}(A, i)$ is not an injective function. The first part of the construction given in the previous proof gives us:

$$\mathcal{C}' = \begin{tikzcd} B \arrow[dr, "f"] & & \\ & C \arrow[r, "i"] & D \\ A \arrow[ur, "g"] & & \end{tikzcd}, \quad \mathcal{M}' = \{i\}, \quad \mathcal{D}' = \{(f, f)\}$$

$\mathrm{Hom}(B, -)$ does not satisfy the disjointness condition. Applying the second part of the construction in the previous proof leads to:

$$\mathcal{C}'' = A \xrightarrow{g} C \xrightarrow{i} D, \quad \mathcal{M}'' = \{i\}, \quad \mathcal{D}'' = \{\}$$

2.13. Proposition. *Let $(\mathcal{C}, \mathcal{M}, \mathcal{D})$ be a normal MD-sketch and let $\mu = (f_i : X \to Y_i)$ be a finite source in $\mathcal{C}$, then it is decidable whether μ is taken to a mono-source in every model of $(\mathcal{C}, \mathcal{M}, \mathcal{D})$.*

An effective decision procedure is contained in the proof below.

Proof. Suppose that μ is taken to a mono-source by every functor in $\mathrm{Mod}(\mathcal{C}, \mathcal{M}, \mathcal{D})$. Then we have the following situation:

$$\mathrm{Mod}(\mathcal{C}, \mathcal{M}) \xleftarrow[J']{} \mathrm{Mod}(\mathcal{C}, \mathcal{M} \cup \{\mu\}) \xleftarrow[J'']{} \mathrm{Mod}(\mathcal{C}, \mathcal{M}, \mathcal{D})$$

where $J = J' \circ J''$ preserves colimits (by lemma 2.10), and where J' and J'' are full inclusions, and hence reflect colimits.

Consider the following diagram in $\mathrm{Mod}(\mathcal{C}, \mathcal{M} \cup \{\mu\})$:

$$\begin{tikzcd} & \mathrm{Hom}(Y_1, -) \arrow[dl, "\mathrm{Hom}(f_1,-)"'] \arrow[dr, "\mathrm{Hom}(f_1,-)"] & \\ \mathrm{Hom}(X, -) & \vdots & \mathrm{Hom}(X, -) \\ & \mathrm{Hom}(Y_n, -) \arrow[ul, "\mathrm{Hom}(f_n,-)"] \arrow[ur, "\mathrm{Hom}(f_n,-)"'] & \end{tikzcd}$$

Its colimit is $\mathrm{Hom}(X, -)$. But since J'' reflects and $J' \circ J''$ preserves this colimit, the colimit of this diagram in $\mathrm{Mod}(\mathcal{C}, \mathcal{M})$ is also $\mathrm{Hom}(X, -)$.

Conversely, if the colimit of that diagram in $\mathrm{Mod}(\mathcal{C}, \mathcal{M})$ is $\mathrm{Hom}(X, -)$, it follows easily that μ will be taken to a mono-source by every functor of $\mathrm{Mod}(\mathcal{C}, \mathcal{M}, \mathcal{D})$.

Hence, we can decide whether μ is taken to a mono-source in every model by computing the colimit of the diagram above in $\mathrm{Mod}(\mathcal{C}, \mathcal{M})$, and checking if this colimit is naturally isomorphic to $\mathrm{Hom}(X, -)$.

Since colimits of finite diagrams in $\mathrm{Mod}(\mathcal{C}, \mathcal{M})$ are computable (first compute the colimit in $\mathrm{Fun}(\mathcal{C}, \mathbf{FinSet})$ and then compute the reflection of this colimit along the inclusion), the result follows. ∎

2.14. PROPOSITION. *Let $(\mathcal{C}, \mathcal{M}, \mathcal{D})$ be a normal MD-sketch and let $\delta = (f_i, g_i)$ be a finite double-source in $\mathcal{C}$, then it is decidable whether δ is taken to a disjoint double-source in every model of $(\mathcal{C}, \mathcal{M}, \mathcal{D})$.*

Again, an effective decision procedure is contained in the proof below.

PROOF. Let (F, ι) be the colimit of the following diagram in $\mathrm{Mod}(\mathcal{C}, \mathcal{M})$:

$$\begin{tikzcd}
& \mathrm{Hom}(Y_1, -) \arrow[dl, "\mathrm{Hom}(f_1,-)"'] \arrow[dr, "\mathrm{Hom}(g_1,-)"] & \\
\mathrm{Hom}(X, -) & \vdots & \mathrm{Hom}(Z, -) \\
& \mathrm{Hom}(Y_n, -) \arrow[ul, "\mathrm{Hom}(f_n,-)"] \arrow[ur, "\mathrm{Hom}(g_n,-)"'] &
\end{tikzcd}$$

Suppose F satisfies all disjointness conditions in $\mathcal{D}$. Then F is a model which does not satisfy δ, and hence δ is not satisfied in all models of $(\mathcal{C}, \mathcal{M}, \mathcal{D})$.

Suppose F does not satisfy all disjointness conditions in $\mathcal{D}$. Given any functor F' which does not satisfy δ, we can easily construct a commutative cocone on the above diagram into F', and hence an arrow $\alpha : F \to F'$. By lemma 2.9, F' can not be a model. Hence, all models of $(\mathcal{C}, \mathcal{M}, \mathcal{D})$ satisfy δ.

Hence, we can decide whether δ is satisfied in all models by computing the colimit of the above diagram in $\mathrm{Mod}(\mathcal{C}, \mathcal{M})$, and checking whether all disjointness conditions in $\mathcal{D}$ are satisfied in the vertex of this colimit. ∎

Recall that a category is skeletal iff no two distinct object are isomorphic, and that it is Cauchy-complete iff every idempotent arrow is split ([7, 8]).

2.15. LEMMA. *For every normal MD-sketch $(\mathcal{C}, \mathcal{M}, \mathcal{D})$, there exists an equivalent normal MD-sketch $(\mathcal{C}', \mathcal{M}', \mathcal{D}')$ where $\mathcal{C}'$ is skeletal and Cauchy-complete. Moreover, there exists an algorithm to compute such an equivalent MD-sketch.*

The proof below describes an algorithm to compute an equivalent skeletal, Cauchy-complete normal MD-sketch.

PROOF. Let $\mathcal{C}'$ be the skeleton of the Cauchy-completion of $\mathcal{C}$, and consider the functor $p \circ i$ where p is the projection on the skeleton and i is the injection in the Cauchy-completion ([7]). Let $\mathcal{M}' = \{p(i(\mu)) \mid \mu \in \mathcal{M}\}$ and let $\mathcal{D}' = \{p(i(\delta)) \mid \delta \in \mathcal{D}\}$. It is easy to verify that $\mathrm{Mod}(\mathcal{C}, \mathcal{M}, \mathcal{D})$ and $\mathrm{Mod}(\mathcal{C}', \mathcal{M}', \mathcal{D}')$ are equivalent, and that $(\mathcal{C}', \mathcal{M}', \mathcal{D}')$ will be normal if $(\mathcal{C}, \mathcal{M}, \mathcal{D})$ was normal. Again, since $\mathcal{C}$ is finite, it is clear that this construction is computable. ∎

2.16. LEMMA. *Every model $M : \mathcal{C} \to \mathbf{FinSet}$ of a normal MD-sketch $(\mathcal{C}, \mathcal{M}, \mathcal{D})$ is a colimit of representable models.*

PROOF. It is well-known that every object in $\mathrm{Fun}(\mathcal{C}, \mathbf{FinSet})$ is a colimit of representable functors ([8]). Moreover, we know that all the representable functors are models (since $(\mathcal{C}, \mathcal{M}, \mathcal{D})$ is normal). From this and the fact that the inclusion of the model-category in $\mathrm{Fun}(\mathcal{C}, \mathbf{FinSet})$ reflects colimits (since it is full and faithful), the result follows. ∎

2.17. LEMMA. *Let $(\mathcal{C}, \mathcal{M}, \mathcal{D})$ be a normal MD-sketch with $\mathcal{C}$ Cauchy-complete, then the following are equivalent:*

1. *M is a representable model.*

2. *The arrows $\epsilon : \mathrm{colimNat}(M, D) \to \mathrm{Nat}(M, \mathrm{colim}D)$, induced by the universality of the colimit, are epis for every diagram D in $\mathrm{Mod}(\mathcal{C}, \mathcal{M}, \mathcal{D})$ for which the colimit exists.*

PROOF. First we show that representable models have property 2. Let H be a Hom-functor. H belongs to the model-category since the given MD-sketch is normal. Consider again diagram 1. Let η be the unit of the adjunction (R, I). Since $\mathrm{Mod}(\mathcal{C}, \mathcal{M})$ is an epi-reflective subcategory (lemma 2.8), every η_X is epi.

$$\begin{aligned}
\mathrm{Nat}(H, \mathrm{colim}D) &= \mathrm{Nat}(IJH, IJ\mathrm{colim}D) \quad \text{(since } I \circ J \text{ is full and faithfull)}\\
&= \mathrm{Nat}(IJH, I\mathrm{colim}JD) \quad \text{(J preserves colimits)}\\
&= \mathrm{Nat}(IJH, \eta_{\mathrm{colim}IJD} \circ \mathrm{colim}IJD)\\
&= \mathrm{Nat}(IJH, \eta_{\mathrm{colim}IJD}) \circ \mathrm{Nat}(IJH, \mathrm{colim}IJD)\\
&= \mathrm{Nat}(IJH, \eta_{\mathrm{colim}IJD}) \circ \mathrm{colimNat}(IJH, IJD)\\
&= \mathrm{Nat}(IJH, \eta_{\mathrm{colim}IJD}) \circ \mathrm{colimNat}(H, D)
\end{aligned}$$

(For the third equality, see the computation of colimits in epi-reflective subcategories in [1]) Since $\eta_{\mathrm{colim}IJD}$ is epi, and since $\mathrm{Nat}(IJH, -)$ preserves epis, we conclude that H has property 2.

Secondly, suppose that M has property 2. We know from lemma 2.16 that $M = \mathrm{colim}D$ where all objects from D are representable models. Let $\beta : D \to M$ be the colimiting cocone. We have the following commutative diagram:

$$\begin{tikzcd}
\mathrm{colimNat}(M,D) \arrow[dr, "\epsilon"] & \\
& \mathrm{Nat}(M,M) \\
\mathrm{Nat}(M,D) \arrow[uu, "\iota"] \arrow[ur, "\mathrm{Nat}(M,\beta)"'] &
\end{tikzcd}$$

with ϵ an epi. So some object H of $\mathrm{Nat}(M, D)$ contains an α such that $\beta_H \circ \alpha = \mathrm{Id}_M$. Hence, M is a splitting of an idempotent of a representable model H. But since every idempotent in $\mathcal{C}$ is split, M must be a representable model. ∎

2.18. Theorem. *Equivalence of MD-sketches is decidable.*

An algorithm to decide equivalence is given in the proof.

Proof. Suppose we are given two MD-sketches $(\mathcal{C}, \mathcal{M}, \mathcal{D})$ and $(\mathcal{C}', \mathcal{M}', \mathcal{D}')$. We may assume that both MD-sketches are normal and that $\mathcal{C}$ and $\mathcal{C}'$ are skeletal and Cauchy-complete, by lemmas 2.12 and 2.15. We claim that the two sketches are equivalent iff there exists an isomorphism $i : \mathcal{C} \to \mathcal{C}'$ such that

1. for each $\mu \in \mathcal{M}$ and for each $\delta \in \mathcal{D}$, $i(\mu)$ is a mono-source in every model of $(\mathcal{C}', \mathcal{M}', \mathcal{D}')$ and $i(\delta)$ is a disjoint double-source in every model of $(\mathcal{C}', \mathcal{M}', \mathcal{D}')$.
2. for each $\mu' \in \mathcal{M}'$ and for each $\delta' \in \mathcal{D}'$, $i^{-1}(\mu')$ is a mono-source in every model of $(\mathcal{C}, \mathcal{M}, \mathcal{D})$ and $i^{-1}(\delta')$ is a disjoint double-source in every model of $(\mathcal{C}, \mathcal{M}, \mathcal{D})$.

Since $\mathcal{C}$ and $\mathcal{C}'$ are finite, we can enumerate all isomorphisms between them, and hence, by propositions 2.13 and 2.14, it follows that equivalence is decidable.

It remains to prove that the condition above is indeed a sufficient and necessary condition for equivalence. It is obvious that the condition is sufficient. We prove that it is necessary. Suppose we have an equivalence between the two model-categories. Since the property mentioned in lemma 2.17 is clearly preserved by equivalence, we know that the equivalence maps representable models to representable models. By the Yoneda lemma and the fact that $\mathcal{C}$ and $\mathcal{C}'$ are skeletal, we conclude that the equivalence between the model-categories induces an isomorphism between $\mathcal{C}$ and $\mathcal{C}'$. It is easy to verify that this isomorphism satisfies the two conditions mentioned above. ∎

3. Data-Specifications

3.1. Definition and Examples.

3.2. Definition. *A* data-specification *is a triple* $(\mathcal{S}, \mathcal{M}, A)$*, where*

1. $\mathcal{S}$ *is a finite category.*

2. $\mathcal{M}$ *is a finite set of sources in* $\mathcal{S}$*.*

3. $A : \mathcal{S}_0 \to \mathbf{FinSet}$ *is a functor, where* $\mathcal{S}_0$ *is the discrete category whose set of objects is the set of objects of* $\mathcal{S}$*.*

Essentially, A is just a function from the objects of $\mathcal{S}$ to the class of finite sets.

3.3. Definition. *A* model *of a data-specification* $(\mathcal{S}, \mathcal{M}, A)$ *is a pair* (M, λ)*, where*

1. $M : \mathcal{S} \to \mathbf{FinSet}$ *is a functor taking every* $\mu \in M$ *to a mono-source.*

2. $\lambda : M \circ I \to A$ *is a natural transformation, where* $I : \mathcal{S}_0 \to \mathcal{S}$ *is the inclusion.*

The following examples are taken from [13]. The category $\mathcal{S}$ of the specification is often given as a graph $\mathcal{G}$, and a set $\mathcal{E}$ of equations. The category $\mathcal{S}$ is defined to be the free category on $\mathcal{G}$, divided by the congruence generated by the equations in $\mathcal{E}$. We begin with a few examples for which the functor A is the constant functor on 1, the terminal set. Such data-specifications are in fact the same as MD-sketches with an empty set of disjointness conditions.

Example. If $\mathcal{S}$ is a discrete category (no non-identity arrows), the models are just typed sets. Let $\mathcal{S}$ be the discrete category with two objects (call them COMPUTER and PRINTER) and no arrows, and let $\mathcal{M}$ be empty. This specification says that the part of the world we want to specify consists of two kinds of entities (computers and printers), and that is all it says.

Example. Arrows in the category specify existential dependencies. Consider for instance:

$$\mathcal{G} = \begin{array}{c} \text{COMPUTER} \\ \downarrow{\scriptstyle l} \\ \text{LOCATION} \end{array}, \quad \mathcal{E} = \emptyset, \quad \mathcal{M} = \emptyset$$

Since the arrow must be taken to a function in a model, this specifies that every computer must have a location associated with it. (An entity of type COMPUTER is always associated with an entity of type LOCATION.)

Example. A source with n arrows in the category can be seen as an n-ary multirelation:

$$\mathcal{G} = \begin{tikzcd} & \text{CONNECTION} \arrow[dl, "c"'] \arrow[dr, "p"] & \\ \text{COMPUTER} & & \text{PRINTER} \end{tikzcd}, \quad \mathcal{E} = \emptyset, \quad \mathcal{M} = \emptyset$$

This specification says that connection is a multirelation between computers and printers: every entity of type CONNECTION is associated with a couple of entities (x, y) with x of type COMPUTER and y of type PRINTER. It is possible that two different entities of type CONNECTION are associated with the same couple. Hence CONNECTION is a *multi*-relation over COMPUTER and PRINTER. For example, a printer could be connected twice to a computer, once with a serial cable, and once with a parallel cable.

Example. A multi-relation is an ordinary relation (no duplicates allowed) if and only if the corresponding source is a mono-source. The specification:

$$\mathcal{G} = \begin{tikzcd} & \text{CONNECTION} \arrow[dl, "c"'] \arrow[dr, "p"] & \\ \text{COMPUTER} & & \text{PRINTER} \end{tikzcd}, \quad \mathcal{E} = \emptyset, \quad \mathcal{M} = \{(\text{CONNECTION},(\text{c},\text{p}))\}$$

says that CONNECTION is an ordinary relation over COMPUTER and PRINTER. A printer can be connected only once to a computer.

Example. By requiring certain equations to be valid in $\mathcal{S}$, we can express equality constraints:

$$\mathcal{G} = \begin{tikzcd} & \text{CONNECTION} \arrow[dl, "c"'] \arrow[dr, "p"] & \\ \text{COMPUTER} \arrow[dr, "l_1"'] & & \text{PRINTER} \arrow[dl, "l_2"] \\ & \text{LOCATION} & \end{tikzcd}, \quad \mathcal{E} = \{l_2 \circ p = l_1 \circ c\}, \quad \mathcal{M} = \emptyset$$

This specification says that computers and printers can be connected only if they have the same location. This kind of constraint occurs very often in practice.

In the examples we have discussed up to now, the functor A was always the constant functor 1. A data-specification with $A = 1$ only specifies the types of entities that exist in the part of the world we want to specify, and some structural constraints (e.g. a computer is always associated with a location, a printer can only be connected to a computer if they share the same location). In a database, we also want to store attributes of the entities. For example, for a printer, we might want to store whether it is a laserprinter or a matrix-printer, and for a computer, we might want to store the type of its processor, or its amount of memory. With the functor A, we specify for each type of entity (each node of $\mathcal{S}$) the set of attribute values that entities of that type can have. Looking at the definition of a model of a data-specification, it is easy to see that in every model M, all entities of type C (i.e. all elements of the set $M(C)$) will be labelled with an element of $A(C)$. The function which takes each entity to its label is the component of the natural transformation λ at C.

Example. Consider for example:

$$
\mathcal{G} = \begin{tikzcd}
& \text{CONNECTION} \arrow[dl, "c"'] \arrow[dr, "p"] & & \\
\text{COMPUTER} \arrow[dr, "l_1"'] & & \text{PRINTER} \arrow[dl, "l_2"] \arrow[r, dotted, no head] & \boxed{\{\text{ matrix, laser }\}} \\
& \text{LOCATION} & &
\end{tikzcd} \qquad (2)
$$

$$\mathcal{E} = \{l_1 \circ c = l_2 \circ p\}, \quad \mathcal{M} = \{(\text{CONNECTION}, (c, p))\}$$

The functor A is defined on the figure, using dotted lines: if there is a dotted line from a node of the graph to a finite set, then this defines the functor A for that node. For example: A(PRINTER) is the set { matrix, laser}. If there is no dotted line from a given node n in the graph, then $A(n)$ is defined to be the terminal set. In a model of this specification, every entity of type PRINTER will be labelled with a value from the set { matrix, laser }. In a similar way, we could have labelled computers with their type of processor, and connections with their data-transmission speed. However, remember that attribute sets must be finite sets.

With these examples in mind, it should be clear that data-specifications provide for an intuitive way to specify the structure of a database. The last example, for instance specifies a (small) database, containing information about the hardware equipment of a company. A model (M, λ) of this data-specification is a possible instance of the database. The functor M indicates how many entities of each type exist, and how they are related to each other. The natural transformation λ gives an attribute value for each entity.

In fact, the most widely used data-specification mechanisms, namely those based on Entity-Relationship diagrams ([10]), are very close to our data-specifications. For more details on how to convert Entity-Relationship diagrams to our data-specifications, consult [13]. That Entity-Relationship diagrams are of major importance in database design can be judged from the fact that an annual international conference on the Entity-Relationship Approach has been organized since 1981. Almost any introductory book on database design includes examples of Entity-Relationship diagrams (see for instance [14] or [16]). Since these diagrams can be reformulated as categorical data-specifications, we refrain from giving more extended examples in this paper, referring the reader to books like [14] and [16] instead.

3.4. Definition. *A* homomorphism *between two models* (M, λ) *and* (M', λ') *of a data-specification* $(\mathcal{S}, \mathcal{M}, A)$ *is a natural transformation* $\alpha : M \to M'$ *such that* $\lambda' \circ \alpha I = \lambda$.

In other words, it is a natural transformation between M and M' that is compatible with the labelling.

Models and homomorphisms of models of a specification $\mathcal{F} = (\mathcal{S}, \mathcal{M}, A)$ form a category, the *model category* of $\mathcal{F}$, which is denoted as $\mathrm{Mod}(\mathcal{F})$. Let $I : \mathcal{S}_0 \to \mathcal{S}$ be the inclusion, let $I^* : \mathrm{Fun}(\mathcal{S}, \mathbf{FinSet}) \to \mathrm{Fun}(\mathcal{S}_0, \mathbf{FinSet})$ be the functor of composition with I and let $\overline{A} : 1 \to \mathrm{Fun}(\mathcal{S}_0, \mathbf{FinSet})$ be the functor picking out A. Then, 3.5. Lemma. *$\mathrm{Mod}(\mathcal{F})$ is a full subcategory of the comma-category $(I^* \downarrow \overline{A})$.*

The proof of this lemma is trivial. An object (F, α) of $(I^* \downarrow \overline{A})$ belongs to $\mathrm{Mod}(\mathcal{F})$ iff the functor F takes all sources in $\mathcal{M}$ to mono-sources.

3.6. Lemma. *Let $A^! : \mathcal{S} \to \mathbf{FinSet}$ be the right Kan extension of A along I. $(I^* \downarrow \overline{A})$ is isomorphic to $\mathrm{Fun}(\mathcal{S}, \mathbf{FinSet})/A^!$.*

Proof. First, note that the right Kan extension exists, since $\mathcal{S}$ is a finite category, and $\mathbf{FinSet}$ is finitely complete. ¿From the universal property of the right Kan extension, we get the natural isomorphism:

$$\mathrm{Nat}(I^*(F), A) \approx \mathrm{Nat}(F, A^!) \tag{3}$$

Objects of $(I^* \downarrow \overline{A})$ are couples (F, λ) with $F : \mathcal{S} \to \mathbf{FinSet}$ a functor and $\lambda : I^*(F) \to A$ a natural transformation. Objects of $\mathrm{Fun}(\mathcal{S}, \mathbf{FinSet})/A^!$ are couples $(F, \overline{\lambda})$ with $F : \mathcal{S} \to \mathbf{FinSet}$ a functor and $\overline{\lambda} : F \to A^!$ a natural transformation. By (3), there is a bijection between the objects of both categories.

Arrows in both categories are natural transformations α. In $(I^* \downarrow \overline{A})$ they must satisfy commutativity of:

$$\begin{tikzcd}
I^*(F_1) \arrow[dd, "I^*(\alpha)"'] \arrow[dr, "\lambda_1"] & \\
& A \\
I^*(F_2) \arrow[ur, "\lambda_2"'] &
\end{tikzcd}$$

In $\mathrm{Fun}(\mathcal{S}, \mathbf{FinSet})/A^!$ they must satisfy commutativity of:

$$\begin{tikzcd}
F_1 \arrow[dd, "\alpha"'] \arrow[dr, "\overline{\lambda}_1"] & \\
& A^! \\
F_2 \arrow[ur, "\overline{\lambda}_2"'] &
\end{tikzcd}$$

However by the naturality in F of (3), these two conditions are equivalent, and hence $(I^* \downarrow \overline{A})$ and $\mathrm{Fun}(\mathcal{S}, \mathbf{FinSet})/A^!$ are isomorphic. ∎

3.7. Equivalence of Data-Specifications. We say that two data-specifications are equivalent iff their model-categories are equivalent. What we want to find is an algorithm to decide equivalence of data-specifications. Our approach will be to translate data-specifications to MD-sketches and then apply theorem 2.18. But first we give an example of equivalent data-specifications.

Example. The following specification is equivalent to specification (2).

$$\mathcal{G} \;=\; \begin{tikzcd}[column sep=small] \text{M-CONNECTION} \arrow[d, "m"] \arrow[dr, "c_1"] & & \text{L-CONNECTION} \arrow[dl, "c_2"'] \arrow[d, "l_1"] \\ \text{MATRIXPRINTER} \arrow[dr, "l_2"] & \text{COMPUTER} \arrow[d, "l_3"'] & \text{LASERPRINTER} \arrow[dl, "l_4"'] \\ & \text{LOCATION} & \end{tikzcd} \tag{4}$$

$$\mathcal{E} = \{l_2 \circ m = l_3 \circ c_1, l_3 \circ c_2 = l_4 \circ l_1\}$$

$$\mathcal{M} = \{(\text{M-CONNECTION}, (m, c_1)), (\text{L-CONNECTION}, (c_2, l_1))\}$$

Informally, it is not too hard to see that the two specifications are semantically equivalent. In the specification above, two different entity types are used for matrix-printers and laser-printers, instead of one entity type with an attribute. Further in this paper, we will prove that the model-categories are indeed equivalent.

The fact that the same real-world situation can be specified in a number of non-isomorphic ways is an important problem in database design and software engineering. Suppose for example, that a number of different data-specifications exist, and that these different specifications overlap partly: certain parts of the real world are specified in more than one of the specifications. It is very likely that these parts are specified differently in each of the specifications, and that makes it hard to combine the given specifications into one large specification. This problem of combining data-specifications is called the *view integration* problem in the database literature, and has been studied extensively. We refer the reader to [6] for a survey. The usual approach is to develop heuristic algorithms which try to identify the equivalent subspecifications, and then leave it to a database designer to decide which subspecifications are indeed equivalent and which are not. A provably correct algorithm (in contrast with a heuristic algorithm) could minimize the amount of work that is left to the database designer. In the sequel of this paper, we will develop such an algorithm, by giving a constructive proof of the decidability of equivalence of data-specifications.

4. Translating data-specifications to MD-sketches

In this section we show that, given an arbitrary data-specification, we can compute an MD-sketch with an equivalent model-category. Since we can decide equivalence of MD-sketches, as a corollary, we find that we can decide equivalence of data-specifications.

4.1. Set-valued functors and discrete opfibrations. It is well-known that the category $\mathrm{Fun}(\mathcal{C}, \mathbf{FinSet})$ is equivalent with the category $\mathbf{FinDof}(\mathcal{C})$ of finite discrete opfibrations over $\mathcal{C}$. (See for instance [5] for a description of the equivalence)

Suppose $F : \mathcal{C} \to \mathbf{FinSet}$ is the functor corresponding to a discrete opfibration (dof) $\Psi : \mathcal{E} \to \mathcal{C}$ under this equivalence. In this section, we investigate under what conditions (for Ψ) the functor F takes a source μ to a mono-source, or a double-source δ to a disjoint double-source.

Let $\mu = (f_i : X \to Y_i)$ be a source in $\mathcal{C}$ and suppose B and B' are nodes of $\mathcal{E}$ such that $\Psi(B) = \Psi(B') = X$. The arrow-lifting property of dofs ensures us the existence of two unique sources $g_i : B \to Z_i$ and $g_i' : B' \to Z_i'$ such that $\Psi(g_i) = \Psi(g_i') = f_i$. We say that $B \sim_\mu^\Psi B'$ iff $Z_i = Z_i'$ for all i. This situation is illustrated in the following picture:

$$\begin{tikzcd}
 & B' \arrow[ddr, "g_1'"'] \arrow[ddrrr, "g_n'"] & & & \\
\mathcal{E} \arrow[dddd, "\Psi"'] & B \arrow[dr, "g_1"'] \arrow[drrr, "g_n"] \arrow[ddd] & & & \\
 & & Z_1 = Z_1' \arrow[d] & \cdots & Z_n = Z_n' \arrow[d] \\
 & & {} & & {} \\
 & X \arrow[dr, "f_1"'] \arrow[drrr, "f_n"] & & & \\
\mathcal{C} & & Y_1 & \cdots & Y_n
\end{tikzcd}$$

It is clear that the functor corresponding to Ψ takes μ to a mono-source iff $B \sim_\mu^\Psi B' \Rightarrow B = B'$.

Now, let $\delta = (f_i : X \to Y_i, f_i' : X' \to Y_i)$ be a double-source in $\mathcal{C}$. The functor corresponding to Ψ takes δ to a disjoint double-source iff there is no double-source $\delta' = (g_i : B \to Z_i, g_i' : B' \to Z_i)$ in $\mathcal{E}$ such that $\Psi(\delta') = \delta$.

4.2. Definition. *Let $(\mathcal{S}, \mathcal{M}, A)$ be a data-specification. Let $A^!$ be the right Kan extension of A along $I : \mathcal{S}_0 \to \mathcal{S}$ and let $\int A^!$ be the category of elements of $A^!$, with associated projection $\Pi : \int A^! \to \mathcal{S}$. We define the* translation *of $(\mathcal{S}, \mathcal{M}, A)$ to be the MD-sketch $(\mathcal{C}^t, \mathcal{M}^t, \mathcal{D}^t)$ where*

1. *$\mathcal{C}^t$ is $\int A^!$.*

2. *$\mathcal{M}^t$ is $\{\mu \mid \Pi(\mu) \in \mathcal{M}\}$.*

3. *$\mathcal{D}^t$ is $\{(f_i : X \to Y_i, g_i : Z \to Y_i) \mid X \neq Z, \Pi(f_i) = \Pi(g_i) \in \mathcal{M}\}$.*

4.3. Theorem. *The model-category of a data-specification $(\mathcal{S}, \mathcal{M}, A)$ is equivalent to the model-category of its translation $(\mathcal{C}^t, \mathcal{M}^t, \mathcal{D}^t)$.*

Proof. First we prove that both model categories are equivalent to full subcategories of $\mathbf{FinDof}(\mathcal{C}^t)$.

For $\mathrm{Mod}(\mathcal{C}^t, \mathcal{M}^t, \mathcal{D}^t)$, this is obvious, since this category is a full subcategory of $\mathrm{Fun}(\mathcal{C}^t, \mathbf{FinSet})$, and we know that $\mathrm{Fun}(\mathcal{C}^t, \mathbf{FinSet})$ is equivalent with $\mathbf{FinDof}(\mathcal{C}^t)$.

For $\mathrm{Mod}(\mathcal{S}, \mathcal{M}, A)$, we know from lemmas 3.5 and 3.6 that this category is a full subcategory of $\mathrm{Fun}(\mathcal{S}, \mathbf{FinSet})/A^!$. Therefor, it is equivalent to a full subcategory of $\mathbf{FinDof}(\mathcal{S})/(\Pi : \int A^! \to \mathcal{S})$. But it is well-known that:

$$\mathbf{FinDof}(\mathcal{S})/(\Pi : \textstyle\int A^! \to \mathcal{S}) \text{ is equivalent with } \mathbf{FinDof}(\textstyle\int A^!)$$

Hence, we conclude that both model categories are equivalent to full subcategories of $\mathbf{FinDof}(\mathcal{C}^t)$. It remains to verify that an object of $\mathbf{FinDof}(\mathcal{C}^t)$ corresponds to a model of $(\mathcal{C}^t, \mathcal{M}^t, \mathcal{D}^t)$ iff it corresponds to a model of $(\mathcal{S}, \mathcal{M}, A)$.

An object $\Psi : \mathcal{E} \to \mathcal{C}^t$ corresponds to a model of $(\mathcal{C}^t, \mathcal{M}^t, \mathcal{D}^t)$ iff the functor $F : \mathcal{C}^t \to \mathbf{FinSet}$ corresponding to Ψ satisfies all monicity conditions in $\mathcal{M}^t$ and all disjointness conditions in $\mathcal{D}^t$. We have seen in paragraph 4.1 what this means in terms of Ψ.

An object $\Psi : \mathcal{E} \to \mathcal{C}^t$ corresponds to a model of $(\mathcal{S}, \mathcal{M}, A)$ iff the functor $F : \mathcal{S} \to \mathbf{FinSet}$ corresponding to the dof $\Pi \circ \Psi$ satisfies all monicity conditions in $\mathcal{M}$.

We prove that these two conditions are equivalent.

1. Suppose $\Psi : \mathcal{E} \to \mathcal{C}^t$ corresponds to a model of $(\mathcal{S}, \mathcal{M}, A)$. Then we know that $B \sim_{\mu}^{\Pi \circ \Psi} B'$ implies that $B = B'$, for all $\mu \in \mathcal{M}$.

   Let μ' be an element of $\mathcal{M}^t$ and suppose $B \sim_{\mu'}^{\Psi} B'$, then $B \sim_{\mu}^{\Pi \circ \Psi} B'$ where $\mu = \Pi(\mu')$, and hence $B = B'$. We conclude that all sources in $\mathcal{M}^t$ are taken to mono-sources.

   Let $\delta' = (f_i' : X' \to Y_i', g_i' : Z' \to Y_i')$ be an element of $\mathcal{D}^t$. Suppose δ' is not satisfied by the functor corresponding to Ψ. This means that there is a double-source $\delta = (f_i : X \to Y_i, g_i : Z \to Y_i)$ such that $\Psi(\delta) = \delta'$. From the definition of the translation we know that $\mu = \Pi(f_i') = \Pi(g_i') \in \mathcal{M}$, and that $X' \neq Z'$. Since $\Psi(X) = X'$ and $\Psi(Z) = Z'$, it follows that $X \neq Z$. Yet, we have that $X \sim_{\mu}^{\Pi \circ \Psi} Z$, which contradicts the fact that Ψ corresponds to a model of $(\mathcal{S}, \mathcal{M}, A)$. We conclude that all double-sources in $\mathcal{D}^t$ are taken to disjoint double-sources.

   Hence, $\Psi : \mathcal{E} \to \mathcal{C}^t$ also corresponds to a model of $(\mathcal{C}^t, \mathcal{M}^t, \mathcal{D}^t)$.

2. Suppose $\Psi : \mathcal{E} \to \mathcal{C}^t$ corresponds to a model of $(\mathcal{C}^t, \mathcal{M}^t, \mathcal{D}^t)$.

   Let $\mu \in \mathcal{M}$ and suppose $B \sim_{\mu}^{\Pi \circ \Psi} B'$. Then $\Psi(B)$ must be equal to $\Psi(B')$, or a disjointness condition in $\mathcal{D}^t$ would be violated. But that means that $B \sim_{\mu'}^{\Psi} B'$ for some μ' with $\Pi(\mu') = \mu$. Hence $B = B'$ and Ψ corresponds to a model of $(\mathcal{S}, \mathcal{M}, A)$.

■

4.4. Corollary. *Equivalence of data-specifications is decidable.*

Proof. Apply theorem 4.3 to compute two equivalent MD-sketches, and then apply theorem 2.18 to decide the equivalence of these two MD-sketches. ■

4.5. Remark. Because of theorem 4.3, which states that any data-specification can be translated to an MD-sketch, the reader might wonder whether data-specifications are now obsolete (subsumed by MD-sketches). We believe that this is not the case. Data-specifications have a very intuitive nature, and are very close to the kind of data-specifications that are used in practice. MD-sketches are cumbersome to work with in practice. For example, the translation of a data-specification to an MD-sketch tends to make the specification much larger, and unreadable to a human reader. On the other hand, MD-sketches are easier to treat mathematically, since they are sketches, and a lot is known about sketches and their model categories. Therefor, we believe that both kinds of specifications have their uses.

We end this section with an example of an equivalence proof of two data-specifications.

Example. It is an easy exercise to translate specifications (2) and (4) to MD-sketches. Specification (4) is already an MD-sketch, since the functor A is the constant functor on 1. To translate specification (2), you must compute the category of elements of the right Kan extension of A along the inclusion of $\mathcal{S}_0$ into $\mathcal{S}$. These constructions are straightforward, algorithms can be found in [8]. The resulting category is isomorphic (call the isomorphism i) to the underlying category of specification (4). The set of double-sources of the translation is empty, and the set of sources contains 2 sources which map under i to the 2 sources in the set of sources of specification (4). Hence, the translations of specifications (2) and (4) are isomorphic as MD-sketches. We conclude that their model-categories are equivalent.

5. Conclusion

We have studied MD-sketches, a specific kind of Finite Sum sketches, and we have proved that equivalence of model categories is algorithmically decidable for this class of sketches. Moreover, our proof was constructive: an algorithm to decide the equivalence of MD-sketches can be extracted from the proof.

Then we have shown that data-specifications, as used in database design, can be translated to MD-sketches. Again, the proof was constructive, and an algorithm to compute the translation was given.

As a consequence, we obtain an algorithm to decide the equivalence of data-specifications. Moreover, the proofs are also constructive in the following sense: if you find that two data-specifications (or two MD-sketches) are indeed equivalent, then the equivalence between the model categories itself is also computable. Given a model of the first specification, it is possible to compute the image of this model under the equivalence, giving you a corresponding model of the second specification.

The ability to decide equivalence of data-specifications, and to compute corresponding models for equivalent specifications is of major importance during view-integration, the process of combining several partly overlapping data-specifications into one big data-specification.

References

[1] J. Adámek, H. Herrlich, G. E. Strecker. *Abstract and Concrete Categories* Wiley-Interscience publications, 1990.

[2] J. Adámek, J. Rosický. *Locally presentable and accessible categories*, Cambridge University Press, 1994.

[3] J. Adámek, J. Rosický. "Finitary sketches" *Journal of Symbolic Logic*, to appear.

[4] J. Adámek, J. Rosický. "On geometric and finitary sketches", preprint.

[5] M. Barr, C. Wells. *Category Theory for Computing Science* Prentice Hall International Series in Computer Science, 1990.

[6] C. Batini, M. Lenzerini, S.B. Navathe. "A comparitive analysis of methodologies for database schema integration" *ACM Computing Surveys*, Vol. 15, nr. 4, 1986, pp. 323–364.

[7] F. Borceux, D. Dejean. "Cauchy Completion in Category Theory" *Cahiers de Topologie et Géometrie Différentielle Catégoriques*, Vol. XXVII–2, pp. 133–146, 1986.

[8] F. Borceux. *Handbook of categorical algebra I* Cambridge University Press, 1993.

[9] B. Cadish, Z. Diskin. "Algebraic graph-based approach to management of multibase systems, I: Schema integration via sketches and equations." To appear in the proceedings of Next Generation of Information Technologies and Systems, NGITS'95, Naharia, Israel, June 1995.

[10] P. P. Chen. "The Entity-Relationship Model – Towards a Unified View of Data" *ACM Transactions on Database Systems*, Vol. 1, No. 1,1976,pp. 9–36.

[11] Z. Diskin, B. Cadish, I.Beylin. "Algebraic graph-based approach to management of multibase systems, II: Algebraic aspects of schema integration." To appear in the proceedings of the Moscow ACM Chapter Conference ADBIS'95, Moscow, Russia, June 1995.

[12] M. Makkai. "Generalized sketches as a framework for completeness theorems." To appear in Journal of Pure and Applied Algebra.

[13] F. Piessens, E. Steegmans. "Canonical forms for data-specifications", Proceedings of Computer Science Logic 94, Springer Verlag LNCS 933, pp. 397–411.

[14] T. J. Teorey. *Database Modeling and Design*, Morgan Kaufmann Publishers, Inc. , 1990.

[15] C. Tuijn. "Data modeling from a categorical perspective", Phd. thesis. University of Antwerpen, 1994.

[16] J. D. Ullman. *Principles of Database and Knowledge-base Systems*, vol. I, Computer Science Press, 1988.

[17] C. Wells. "Sketches: outline with references".

Dept. of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
Email: `Frank.Piessens@cs.kuleuven.ac.be`

This article may be accessed via WWW at `http://www.tac.mta.ca/tac/` or by anonymous ftp at `ftp://ftp.tac.mta.ca/pub/tac/html/volumes/1995/v1n8.{dvi,ps}`

THEORY AND APPLICATIONS OF CATEGORIES (ISSN 1201-561X) will disseminate articles that significantly advance the study of categorical algebra or methods, or that make significant new contributions to mathematical science using categorical methods. The scope of the journal includes: all areas of pure category theory, including higher dimensional categories; applications of category theory to algebra, geometry and topology and other areas of mathematics; applications of category theory to computer science, physics and other mathematical sciences; contributions to scientific knowledge that make use of categorical methods.

Articles appearing in the journal have been carefully and critically refereed under the responsibility of members of the Editorial Board. Only papers judged to be both significant and excellent are accepted for publication.

The method of distribution of the journal is via the Internet tools WWW/gopher/ftp. The journal is archived electronically and in printed paper format.

Subscription information. Individual subscribers receive (by e-mail) abstracts of articles as they are published. Full text of published articles is available in .dvi and Postscript format. Details will be e-mailed to new subscribers and are available by WWW/gopher/ftp. To subscribe, send e-mail to tac@mta.ca including a full name and postal address. For institutional subscription, send enquiries to the Managing Editor, Robert Rosebrugh, rrosebrugh@mta.ca.

Information for authors. The typesetting language of the journal is TEX, and LATEX is the preferred flavour. TEX source of articles for publication should be submitted by e-mail directly to an appropriate Editor. They are listed below. Please obtain detailed information on submission format and style files from the journal's WWW server at URL http://www.tac.mta.ca/tac/ or by anonymous ftp from ftp.tac.mta.ca in the directory pub/tac/info. You may also write to tac@mta.ca to receive details by e-mail.