Schema Equivalence as a Counting Problem

H.A. (Erik) Proper and Th.P. van der Weide

Institute for Computing and Information Sciences,
Radboud University Nijmegen,
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands, EU
{E.Proper, Th.P.vanderWeide}@cs.ru.nl

Abstract. In this paper we introduce some terminology for comparing the expressiveness of conceptual data modeling techniques, such as ER, NIAM, PSM and ORM, that are finitely bounded by their underlying domains. Next we consider schema equivalence and discuss the effects of the sizes of the underlying domains. This leads to the introduction of the concept of finite equivalence, which may serve as a means to a better understanding of the fundamentals of modeling concepts (utility). We give some examples of finite equivalence and inequivalence in the context of ORM.

1 Schema Equivalence

When modeling a Universe of Discourse ([ISO87]), it is generally assumed that we can recognize *stable states* in this Universe of Discourse, and that there are a number of actions that result in a change of state (*state transitions*). This is called the state-transition model. Furthermore we assume that the Universe of Discourse has a unique *starting state*.

In mathematical terms, a Universe of Discourse U∘D consists of a set $\mathcal{S}$ of states, a binary transition relation τ over states, and an initial state $s_0 \in \mathcal{S}$:

$$\text{U}\circ\text{D} = \langle \mathcal{S}, \tau, s_0 \rangle$$

The purpose of the modeling process is to construct a formal description, (a *specification*) Σ of U∘D, in terms of some underlying formalism. This specification will have a component $\mathcal{S}(\Sigma)$ that specifies $\mathcal{S}$, a component $\tau(\Sigma)$ that specifies τ, and a state $s_0(\Sigma)$ that is designated as the initial state s_0.

The main requirement for specification Σ is that it behaves like U∘D. This can be shown by a (partial) function h, relating the states from $\mathcal{S}(\Sigma)$ to the (real) states $\mathcal{S}$ from U∘D such that h shows this similarity. Such a function is called a *(partial) homomorphism*. If each state of U∘D is captured by the function h, we call Σ a *correct specification* with respect to U∘D, as each state of U∘D has a representation in Σ. In that case, the function h is surjective, and called an *epimorphism* (see also [Bor78]).

Definition 1. *We call h a* partial *homomorphism between Σ and U∘D if*

1. *h is a (partial) function $h : \mathcal{S}(\Sigma) \rightarrow \mathcal{S}$*

R. Meersman et al. (Eds.): OTM Workshops 2005, LNCS 3762, pp. 730–739, 2005.

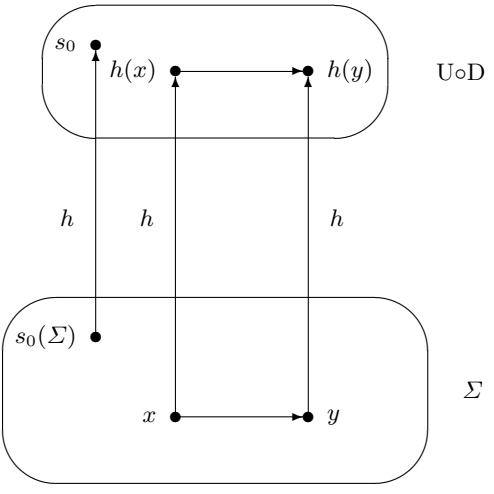

Fig. 1. A correct specification

2. transitions commute under h:

$$\forall_{s,t \in \mathcal{S}(\Sigma)} \left[\langle s,t \rangle \in \tau(\Sigma) \Leftrightarrow \langle h(s), h(t) \rangle \in \tau \right]$$

3. h maps the initial state of the specification onto the initial state of U∘D:

$$h(s_0(\Sigma)) = s_0$$

If h is surjective, we call h an epimorphism between Σ and U∘D.

We call an algebra $\mathcal{A}$ (partially) homomorphic with algebra $\mathcal{B}$, if there exists a (partial) homomorphism from $\mathcal{A}$ into $\mathcal{B}$. If schema Σ is a description of $\mathcal{A}$, then we will also call Σ (partially) homomorphic with $\mathcal{B}$. The notion of epimorphism is extended analogously.

Note that in a correct specification Σ, a state of U∘D may have more than one corresponding state in $\mathcal{S}(\Sigma)$. In that case we have a redundant representation for the states of U∘D. Redundant representations are useful as they provide opportunities for improvement of efficiency.

The disadvantage of a redundant representation is that we do not have a description of U∘D that is free of implementation (representation) details. A description can only be implementation independent if each state has a unique representant. Such a description is called a *conceptual schema* in the context of information systems. This is the case if the function h that relates Σ to U∘D is bijective.

The *expressiveness* of a formal method $\mathcal{M}$ is introduced as the set of "U∘D"'s it can model. This can be described by:

$$\left\{ \langle \mathcal{S}(\Sigma), \tau(\Sigma), s_0(\Sigma) \rangle \;\middle|\; \Sigma \in \mathcal{L}(\mathcal{M}) \right\}$$

If we restrict ourselves in this definition to $\tau(\Sigma) = \emptyset$, we get the so-called *base expressiveness* of method $\mathcal{M}$. The base expressiveness usually is the criterion that is used intuitively when comparing different methods.

From the above definition of conceptual schema, the following definition of schema equivalence can be derived.

Definition 2. *Two specifications Σ and Σ' are equivalent, $\Sigma \cong \Sigma'$, if there exists a homomorphism h from Σ onto Σ' such that h is a bijection.*

2 Schema Equivalence in ORM

In this section we consider the base expressiveness of ORM. We focus on the formal definition as for example specified in the Predicator Model (PM, see [BHW91]), and discuss schema equivalence in that context. Within that context, we may omit differences between ORM and PM.

Let Σ be an ORM schema, with underlying label type set $\mathcal{L}$, then this schema specifies the following set of states:

$$\mathcal{S}(\Sigma) = \{p \mid \mathsf{IsPop}_{\mathcal{L}}(\Sigma, p)\}$$

A population p is a function assigning a set of instances to each object type in schema Σ. The $\mathsf{IsPop}_{\mathcal{L}}$ predicate determines whether p is a proper population. The population of label values is restricted to values of some domain $\mathcal{D}$. We will show that the base expressiveness strongly depends on the actual choice of $\mathcal{D}$. In this restricted sense the resulting state space of schema Σ is:

$$\mathcal{S}_{\mathcal{D}}(\Sigma) = \{p \mid \mathsf{IsPop}_{\mathcal{L}}(\Sigma, p) \wedge \forall_{x \in \mathcal{L}} [p(x) \subseteq \mathcal{D}]\}$$

Please note that we restrict ourselves to finite populations, i.e., populations where each object type x has assigned a finite population ($p(x) < \infty$). Using this definition we introduce the notion of domain equivalence.

Definition 3. *Two ORM schemas Σ and Σ' are domain equivalent over domain $\mathcal{D}$, denoted as $\Sigma \cong_{\mathcal{D}} \Sigma'$, if they have equivalent state spaces:*

$$\mathcal{S}_{\mathcal{D}}(\Sigma) \cong \mathcal{S}_{\mathcal{D}}(\Sigma')$$

The sets A and B are called equivalent ($A \cong B$) if there exists a bijection from A into B.

Note that being equivalent does not mean that both schemata are as suitable in representing U∘D. For suitability we should take the complexity of the transition into account. This complexity however is outside the scope of this paper.

A first result is that the underlying domain may be such expressive that any two schemata based on that domain are equivalent:

Lemma 1. *Let Σ and Σ' be ORM schemas then:*

$$\mathcal{D} \text{ countably infinite } \Rightarrow \Sigma \cong_{\mathcal{D}} \Sigma'$$

Proof: We will only give a brief outline of this proof. The important step is to prove that the number of populations in a schema with a countable domain is countable itself (assuming finite populations). This however, is true because every population can be coded as a finite string by ordering the object types in the schema at hand and listing their populations sequentially, according to this ordering, separated by special separator symbols. Each such finite string can uniquely be translated to a finite bitstring, which can be considered as a natural number in binary representation. □

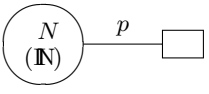

Fig. 2. The most simple universal schema

We conclude that the expressiveness of ORM data structuring in the context of a countably infinite domain is limited as all schemata are equivalent in that case. Note that, in the context of countably infinite domains, this property holds for most other data models as well. Each schema thus can be considered as a universal schema, as it is expressive enough to "simulate" any other schema. The analogon of a universal schema in the algorithmic world is the universal Turing machine (see for example [CAB+72]). The most simple universal schema is shown in figure 2. This simple schema can represent any population of any other schema, by using the counting schema described in the proof above. The role of the unary fact type is to exclude all elements from N that do not correspond to a valid population of the simulated schema.

We restrict ourselves to a finite domain for label values. As a direct consequence, schema Σ has a finite state space. We introduce the notion of finite equivalence:

Definition 4. *Two ORM schemata Σ and Σ' are finite equivalent denoted as $\Sigma \cong_f \Sigma'$, if they have an equivalent state space for equivalent finite underlying domains, or if for all $\mathcal{D}$ and $\mathcal{D}$':*

$$\mathcal{D} \cong \mathcal{D}' \wedge |\mathcal{D}| < \infty \Rightarrow \mathcal{S}_{\mathcal{D}}(\Sigma) \cong \mathcal{S}_{\mathcal{D}'}(\Sigma')$$

Finite equivalence can be proven by the construction of a bijection between the two state spaces of the schemas.

Example 1. The schemas Σ and Σ' from figure 3, are finite equivalent.

Proof: *The basic idea is to define a translation from instances from Σ to instances from Σ' such that we have a bijection between $\mathcal{S}(\Sigma)$ and $\mathcal{S}(\Sigma')$. This is achieved by relating identical instances of object types A, B and C in both schemas and instances $\{p : a, q : b\}$ in $\mathsf{Pop}(f)$ and $\{r : \{p : a, q : b\}, s : c\}$ in $\mathsf{Pop}(g)$ to one instance $\{t : a, u : b, v : c\}$ in $\mathsf{Pop}(h)$.*

Note the importance of the total role (the black dot) on predicator r in this transformation. Its semantics is:

$$x \in \mathsf{Pop}(f) \Rightarrow \exists_{y \in \mathsf{Pop}(g)} [y(r) = x]$$

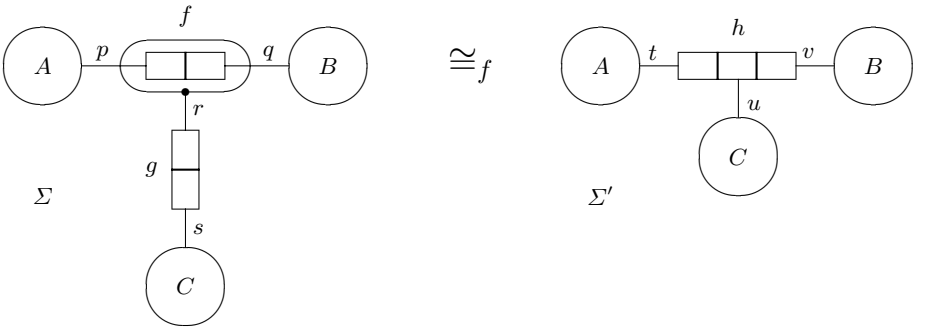

Fig. 3. Example of finite equivalence

Therefore, the total role makes it unnecessary to consider instances of fact type f that do not contribute in fact type g. For a general definition of the semantics of constraints in NIAM schemas, refer to [BHW91]. □

Finite inequivalence can be proven by showing that the state spaces of the underlying schemas are not equal in size.

Example 2. If we omit the total role from schema Σ in figure 3, the schemas are not *finite equivalent.*

Proof: *Let a, b and c be the population size of A, B and C respectively. The number of populations of fact type f amounts to:*

$$\sum_{i=0}^{ab} \binom{ab}{i} = 2^{ab}$$

Now suppose f is populated with i tuples, then for g we can have 2^{ic} *different populations. The number of populations of Σ therefore amounts to:*

$$\begin{aligned} \sum_{i=0}^{ab} \binom{ab}{i} 2^{ic} &= \sum_{i=0}^{n} \binom{ab}{i} (2^c)^i \\ &= (1+2^c)^{ab} \end{aligned}$$

On the other hand, the number of populations of Σ′ equals $2^{abc} = (2^c)^{ab}$. □

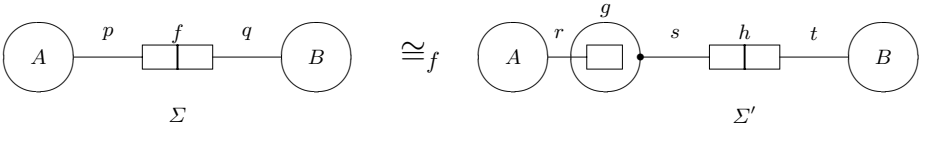

Fig. 4. Another example of finite equivalence

Example 3. In figure 4, another example of finite equivalence is shown.

Proof: *The main observation is that instances occurring in predicator p of schema Σ are to be mapped onto identical instances in the population of fact type g in schema Σ'. Instances of object types A and B in both schemas are again related via an identical mapping. Instances in fact type f in schema Σ are related to identical instances in fact type h in schema Σ'.* □

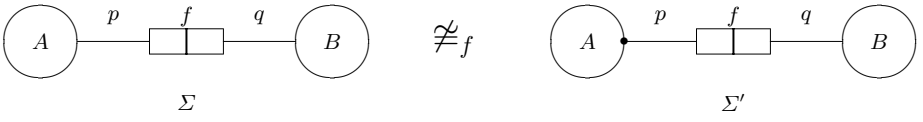

Fig. 5. Example of finite inequivalence

Example 4. In figure 5 two schemas are depicted, which are not finite equivalent.

Proof: *It is not hard to see that the number of populations in Σ with $|\mathsf{Pop}(A)| = a$ and $|\mathsf{Pop}(B)| = b$ is $(2^b)^a$, while the number of populations in Σ' with the same restriction is $(2^b - 1)^a$.* □

3 An Upper Bound for Populatability

A data modeling technique is called finitely bounded by its underlying domains, if each schema from that technique allows for a finite number of populations, in case of a finite domain of label values.

Definition 5. *The populatability of a schema Σ is:*

$$m_{\mathcal{D}}(\Sigma) = \left\| \mathcal{S}_{\mathcal{D}}(\Sigma) \right\|$$

As each schema can be populated by the empty population ([BHW91]), an immediate consequence is:

Lemma 2.

$$\|\mathcal{D}\| = 0 \Rightarrow \forall_{\Sigma \in \mathcal{L}(\mathcal{M})} \left[m_{\mathcal{D}}(\Sigma) = 1 \right]$$

Definition 6. *Method $\mathcal{M}$ is called finitely bounded by its underlying domains $\mathcal{D}$ if:*

$$\|\mathcal{D}\| < \infty \Rightarrow \forall_{\Sigma \in \mathcal{L}(\mathcal{M})} \left[m_{\mathcal{D}}(\Sigma) < \infty \right]$$

In this section we derive an upper bound on the populatability of a schema. In order to simplify the derivation, we restrict ourselves to fact schemata, i.e., schemata Σ without entity types (i.e., $\mathcal{E}(\Sigma) = \emptyset$).

Lemma 3.

$$\forall_{\Sigma} \exists_{\Sigma'} \left[\Sigma \equiv \Sigma' \wedge \mathcal{E}(\Sigma') = \emptyset \right]$$

Proof: Replace each entity type by a fact type, corresponding to its identification. If the identification of entity type x consists of the convolution of k path expressions (i.e., $mult(x) = k$, see [HPW93]), then this replacement leads to the introduction of a k-ary fact type. The resulting schema is denoted as $de(\Sigma)$. Then obviously $\Sigma \equiv de(\Sigma)$ and $\mathcal{E}(de(\Sigma)) = \emptyset$. □

The number $p(de(\Sigma))$ of predicators of schema $de(\Sigma)$ is found by:

Lemma 4.

$$p(de(\Sigma)) = p(\Sigma) + \sum_{x \in \mathcal{E}(\Sigma)} mult(x)$$

Proof: Obvious!

Next we introduce a series $\{N_p\}_{p \geq 0}$ of schemata (see figure 6), consisting of a single p-ary fact type over some label type L. These schemata are the best populatable schemata among schemata with the same number of predicators. □

Theorem 1.

$$\|\mathcal{D}\| > 1 \Rightarrow \forall_{\Sigma} \left[m(\Sigma) \leq m(N_{p(de(\Sigma))}) \right]$$

Proof: First we remark $m(\Sigma) = m(de(\Sigma))$. Next we use the fact that a schema becomes better populatable by undeeper nesting of (at least) binary fact types. This is shown in lemma 5. Furthermore, merging fact types improves populatability (see lemma 7). By repeatedly applying these steps, schema $N_{p(de(\Sigma))}$ will result. □

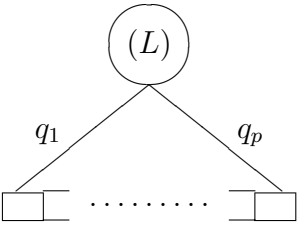

Fig. 6. Best populatable schemata

Lemma 5. *Consider the schemata Σ_1, Σ_2 and Σ_3 from figure 7, then:*

$$\|\mathcal{D}\| > 1 \Rightarrow m(\Sigma_1) \leq m(\Sigma_2) \leq m(\Sigma_3)$$

Table 1. Growth of populatability

n	$m(\Sigma_1)$	$m(\Sigma_2)$	$m(\Sigma_1)$
0	1	1	1
1	3	3	2
2	21	81	256
3	567	19683	134217728
4	67689	43046721	1.845E+19

Proof: Let $\|\mathcal{D}\| = n$, then:

$$\begin{aligned} m(\Sigma_1) &= \sum_{i=0}^{n} \binom{n}{i} \sum_{j=0}^{i} \binom{i}{j} 2^{(j^2)} \\ &\leq \sum_{i=0}^{n} \binom{n}{i} \sum_{j=0}^{i} \binom{i^2}{j^2} 2^{(j^2)} \\ &\leq \sum_{i=0}^{n} \binom{n}{i} \sum_{j=0}^{i^2} \binom{i^2}{j} 2^{j} = m(\Sigma_2) \\ m(\Sigma_2) &= \sum_{i=0}^{n} \binom{n}{i} \sum_{j=0}^{i^2} \binom{i^2}{j} 2^{j} \\ &= \sum_{i=0}^{n} \binom{n}{i} 3^{(i^2)} \\ m(\Sigma_3) &= \sum_{i=0}^{n} \binom{n}{i} 2^{(i^3)} \end{aligned}$$

The result follows from the observation:

$$n > 1 \Rightarrow 2^{(n^3)} > 3^{(n^2)}$$

The populatability of schemata $\{N_p\}_{p \geq 0}$ grows extremely fast. □

Lemma 6.

$$m(N_p) = \sum_{i=0}^{n} \binom{n}{i} 2^{(i^p)}$$

Lemma 7.

$$m(N_p) * m(N_q) \leq m(N_{p+q})$$

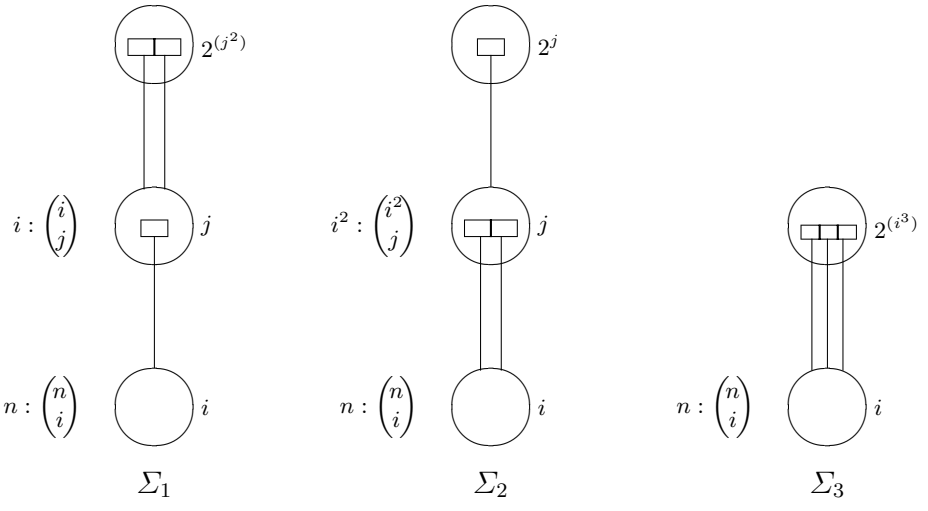

Fig. 7. Transformation steps

From theorem 1 we conclude that ER, NIAM and ORM are finitely bounded by their underlying domains.

4 Conclusions

In this paper we introduced some fundamental notions for the expression and comparison of the expressive power of conceptual schemata. In practise this will not be very helpful. However, by gaining a deeper understanding of the basic limitations of modeling techniques, we may be better equipped to improve state-of-the-art techniques.

Acknowledgements

The authors wish to thank Arthur ter Hofstede for his involvement in this paper.

References

[BHW91] P. van Bommel, A.H.M. ter Hofstede, and Th.P. van der Weide. Semantics and verification of object-role models. *Information Systems*, 16(5):471–495, October 1991.

[Bor78] S.A. Borkin. Data Model Equivalence. In *Proceedings of the Fourth International Conference on Very Large Data Bases*, pages 526–534, 1978.

[CAB+72] J.N. Crossley, C.J. Ash, C.J. Brickhill, J.C. Stillwell, and N.H. Williams. *What is mathematical logic?* Oxford University Press, Oxford, United Kingdom, 1972.

[HPW93] A.H.M. ter Hofstede, H.A. (Erik) Proper, and Th.P. van der Weide. Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 18(7):489–523, October 1993.

[ISO87] *Information processing systems – Concepts and Terminology for the Conceptual Schema and the Information Base*, 1987. ISO/TR 9007:1987. `http://www.iso.org`