Data Modelling in Complex Application Domains

A.H.M. ter Hofstede [1] H.A. Proper [2] Th.P. van der Weide [3]

PUBLISHED AS:

A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Data Modelling in Complex Application Domains. In P. Loucopoulos, editor, *Proceedings of the Fourth International Conference CAiSE'92 on Advanced Information Systems Engineering*, volume 593 of *Lecture Notes in Computer Science*, pages 364–377, Manchester, United Kingdom, EU, May 1992. Springer Verlag, Berlin, Germany, EU. ISBN 3540554815

Abstract

In many non trivial application domains, object types with a complex structure occur. Data modelling techniques which only allow flat structures are not suitable for representing such complex object types. In this paper a general data modelling technique, the Predicator Set Model, is introduced, which is capable of representing complex structures in a natural way.

The expressiveness of the Predicator Set Model is illustrated by means of a number of examples. In those examples, the Predicator Set Model's expressiveness is related to the expressiveness of more traditional modelling techniques. Furthermore, some notational conventions are defined, which enable a more compact representation of complex structures.

1 Introduction

The conventional Relational Model and ER approach allow for a high-level description of data and relations, abstracting from representation and implementation details. Main disadvantage, however, is their incapability of representing complex structures in a natural way. In these techniques, complex structures have to be "flattened", i.e. represented non-hierarchically, which leads to overspecification. This in turn does not comply with the *conceptualisation principle* as it is formulated in [ISO87].

Various application domains indeed contain objects with complex structures. Documents (and Hypertexts) are an example in the field of office automation. In [Wig90] it is estimated that 1% of all recorded information is contained in so-called formatted databases (e.g. a relational database), 4% is recorded on microfiche, while the remaining 95% is contained in unformatted databases. Unformatted databases are capable of containing objects with variable components and varying size (e.g. documents and graphics; typically grammar governed data).

Another domain in which complex objects are important is the field of method engineering or meta-modelling ([VHW91]). In this field, meta-models are constructed, capturing the structure of models that are expressed in some modelling technique. Many modelling techniques contain concepts that correspond to complex structures, e.g. whole diagrams have such a complex structure. It is not natural to represent these object types as flat structures (see e.g. [Wel88]). Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) are also areas in which such complex structures frequently occur.

Finally, in the development of so termed Evolving Information Systems ([FOP92], [MS90], [Ari91], [Rod91]) where the information structure itself is allowed to change over time as well, there is a need

[1] This work has been partially supported by SERC project SOCRATES. Software Engineering Research Centre (SERC), P.O. Box 424, 3500 AK Utrecht, The Netherlands

[2] The investigations were partly supported by the Foundation for Computer Science in the Netherlands (SION) with financial support from the Netherlands Organization for Scientific Research (NWO), University of Nijmegen, E-mail: E.Proper@acm.org

[3] University of Nijmegen, Toernooiveld, 6525 ED Nijmegen, The Netherlands

for a modelling technique which incorporates all basic modelling concepts. This implies the need for a modelling technique with an expressivity which is based on a set of powerful modelling concepts.

In this paper a general data modelling technique is introduced, which has been defined formally in [HW93] and [HPW93]. This modelling technique, the *Predicator Set Modelling technique* (*Predicator Set Model* for short), indeed is capable of representing complex structures in a natural way. In this paper a number of examples are given to make it plausible, from a practical point of view, that the Predicator Set Model allows for the elegant representation of complex object types. Notational conventions will be introduced that allow for compact representations of complex objects.

2 Basic Data Modelling Concepts

One of the key concepts in data modelling is the concept of relation type or fact type. In ER ([Che76]) and NIAM ([NH89]) a relation type is considered to be an association between object types. In figure 1 the graphical representation of a binary relation type R between object types X_1 and X_2 in the NIAM style is shown, while in figure 2 the corresponding ER diagram is depicted.

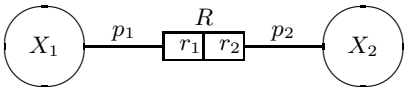


Figure 1: A NIAM relation type

The basic building element of fact types is the connection between an object type and a role, the so-called *predicator*. In figure 1, p_1 is the predicator connecting X_1 to r_1, and p_2 the predicator that connects X_2 to r_2. In the Predicator Set Model, which is an extension of the Predicator Model ([BHW91], [HW92]), a fact type is considered to be a set of predicators. A relation type is therefore considered as an association between predicators, rather than between objects types. Fact types are regarded as object types. This is called objectification. In the sequel some examples of objectifications are shown. Sometimes, we will prefer to denote the predicators involved in a relation type seperately. In section 5 some examples of this are shown.

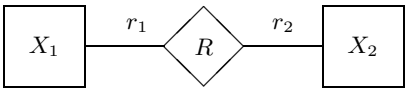


Figure 2: The corresponding ER diagram

Two special kinds of object types are entity types and label types. The difference is that labels can, in contrast with entities, be represented (reproduced) on a communication medium. As a result, label types are also called *concrete* object types. All other object types are called *abstract*, they are not representable by themselves. As usual, a clear distinction is made between concrete object types and abstract object types. The gap between these concrete and abstract object types can only be crossed by special binary fact types. These fact types correspond to *bridge types* in NIAM ([NH89], [Win90]), and *attribute types* in ER ([Che76]). Each entity type must be identifiable in terms of label types.

Anothor basic concept of data modelling is specialisation, also referred to as subtyping. Specialisation is a mechanism for representing one or more (possibly overlapping) subtypes of a type. Intuitively a specialisation relation between a subtype and a supertype implies that the instances of the subtype are also instances of the supertype. For proper specialisation, it is required that subtypes be defined in terms of one or more of their supertypes. Such a decision criterion is referred to as the *Subtype Defining Rule* (see e.g. [BHW91]). Identification of subtypes is derived from their supertypes.

Specialisation relations are organised in so-called specialisation "hierarchies". A specialisation hierarchy is in fact not a hierarchy in the strict sense, but an acyclic directed graph with a unique top. A

specialisation hierarchy can thus be considered a semi-lattice: for each pair of subtypes (in the same hierarchy), the least upper bound should exist. The least upper bound of two subtypes is that object type that is supertype of both subtypes, and that has no subtype with this property. The top of this semi-lattice, i.e. the top of a specialisation hierarchy, will be referred to as the *pater familias* (see [DMV88]). Consequently, the identification of every object type in the hierarchy is derived from the pater familias of the hierarchy.

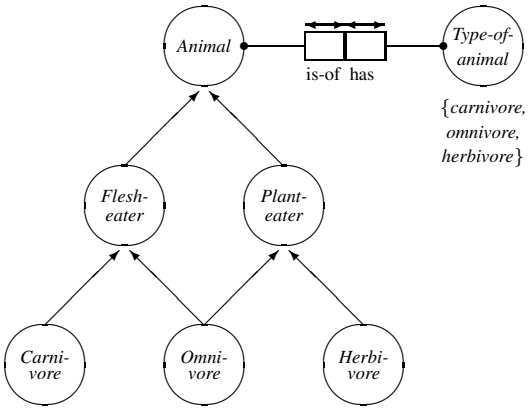


Figure 3: Example of a specialisation hierarchy

As an example of a specialisation hierarchy, consider figure 3. There the following hierarchy is depicted:

Flesh-eater Spec Animal
Plant-eater Spec Animal
Carnivore Spec Flesh-eater
Omnivore Spec Flesh-eater
Omnivore Spec Plant-eater
Herbivore Spec Plant-eater

Each specialisation relation is represented as an arrow in figure 3. As a consequence, the pater familias of object type *Carnivore* is *Animal*. The subtype defining rules are:

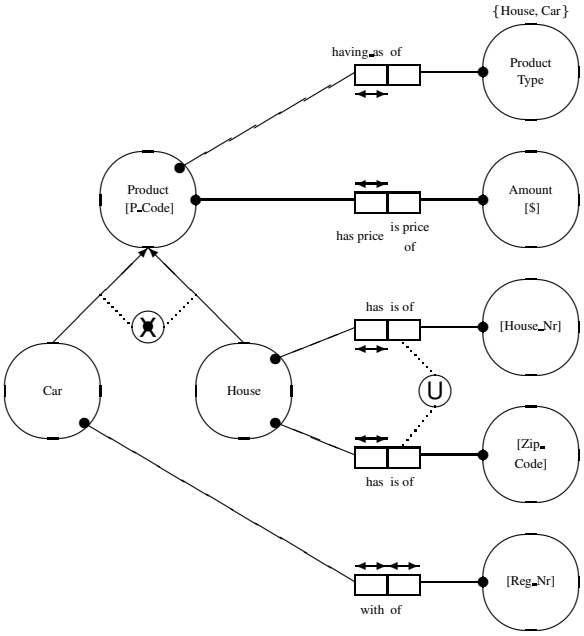
Flesh-eater = Animal is-of Type-of-animal {carnivore, omnivore}
Plant-eater = Animal is-of Type-of-animal {herbivore, omnivore}
Carnivore = Animal is-of Type-of-animal {carnivore}
Omnivore = Animal is-of Type-of-animal {omnivore}
Herbivore = Animal is-of Type-of-animal {herbivore}

3 Generalisation

Generalisation is a mechanism that allows for the creation of a new object type as a generic type for other object types. The constituent object types in a generalisation are called the specifiers of the generalised object type. As a result, the generalised object type is covered by its constituent object types. This means that every instance of any specifier is also an instance of the generalised object type. Another consequence is that the identification of a generalised object type is determined by the identification of its specifiers.

As an example, for the motivation and use of generalisation, consider a pricelist for individually priced *Product*s. A *Product* is either a *Car*, or a *House*. A *Car* is identified by a registration number, while a *House* is identified by the combination of its zip-code and house number. *Product* is thus considered to be a generic term for *House* and *Car*. *Product*s have a price associated to them.

In traditional data modelling techniques, e.g. NIAM and ER, this Universe of Discourse is modelled by the schema in figure 4. Note that the uniqueness of the combination between a zip code and a house



Subtype defining rules:

Car = Product having_as Product_Type 'Car'
House = Product having_as Product_Type 'House'

Figure 4: Subtyping instead of Generalisation

number is modelled by means of an encircled U, a so-called *uniqueness constraint*. For the semantics of complex uniqueness constraints, see [WHB92].

We will point out that this schema suffers from overspecification. Firstly, a special label type (*P_code*) has to be introduced in order to identify *Product*s. Secondly, a special fact type and a special type (*Product Type*) type are required to determine the type of the *Product*. This determination forms the *Subtype Defining Rule* for *Product*s (see figure 4). However, these extra object types are not conceptually relevant. Their introduction should therefore be considered as a violation of the *Conceptualisation Principle* (see [ISO87], [NH89] or [Win90]).

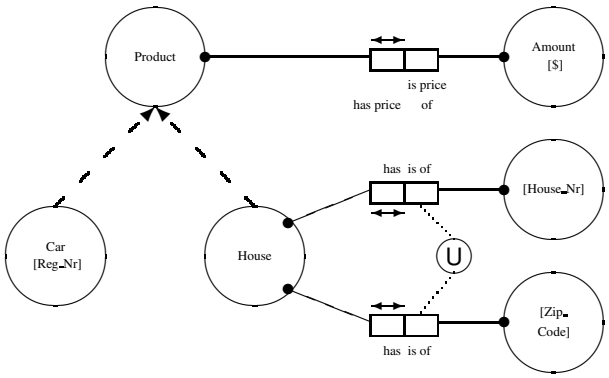


Figure 5: Example of Generalisation

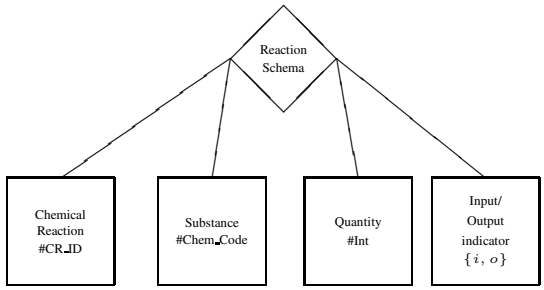
Using the concept of generalisation, these overspecifications are avoided. In figure 4 a more appropriate schema for this Universe of Discourse is depicted. In this schema, the label type *P_code* is no longer needed, since products inherit their identification from *Car*s and *House*s.

4 Power and Sequence Types

Another situation that invites a system analyst, using a conventional modelling technique as eg ER or NIAM, to a violation of the *Conceptualisation Principle*, is when groups of objects occur just as groups, without any other identification than their composition. The most primitive manifestations of this phenomenon are the power set mechanism in formal set theory, and the sequencing (lists) mechanism. However, in its full glory, this phenomenon is a mechanism for *schema decomposition* (see also section 6).

In the Predicator Set Model the concept of *power type* is introduced as the equivalence of power sets in formal set theory. An instance of a power type is a set of instances of its *element type*. An instance of the power type is identified by corresponding instances of the element type, just as a set is identified by its elements in formal set theory (axiom of extensionality), see [HK87].

As an illustration of the expressive power of power object types the chemical reactions example from [Fal93] is discussed. The considered Universe of Discourse deals with simple chemical reactions. A chemical reaction takes a set of input substances with their associated quantities, and produces a set of output substances in corresponding quantities.



R1	H_2	2	i
R1	O_2	1	i
R1	H_2O	2	o
R2	$CH_3 - CH_2OH$	2	i
R2	O_2	1	i
R2	$CH_3 - CHO$	2	o
R2	H_2O	2	o

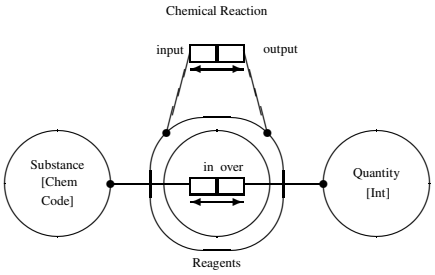
Figure 6: Chemical Reactions in ER

This Universe of Discourse could be modelled in an ER schema in terms of a quartenary relationship, as shown in figure 6. In this relation, the attribute *CR_ID* is used to identify chemical reactions. The entity type *Substance* describes wich substance is subject to the chemical reaction, and the entity type *Quantity* describes in what quantity. The *Input/Output indicator* makes the distinction between input and output substances to the chemical reaction. A first problem with this solution is the superfluous identification of a chemical reaction. Only some chemical reactions are sufficiently important, to have a name of their own. The others are just identified by their description in terms of what goes in and what comes out. The second problem is that this solution does not allow for the addition of a chemical reaction by one elementary update. This is caused by the fact that in the model of 6 several object instances are needed to denote one reaction.

The use of a power type offers a much better opportunity to model this Universe of Discourse (see figure 7). In this model, a chemical reaction is modelled as a relationship between a set of input reagents, and a set of output reagents. This schema is better understood by studying a sample population (see figure 7). This sample population is in the style of nested relations as, encountered in the NF^2 datamodel [SS86]. The main difference is that NF^2 uses a nested table heading (and thus nested tuples).

The solution of figure 7 also solves the update problem which was mentioned before. In this model a chemical reaction is denoted as a single object instance. Therefore, the above mentioned elementary update problem is solved. The consequence is that an update operation of a chemical reaction can be considered as a single operation in the Predicator Set Model.

Sequence types are ordered power types, their instances are tuples of arbitrary length. As an example,



input	*output*
over | *in* H_2 | 2 O_2 | 1	*over* | *in* H_2O | 2
over | *in* $CH_3 - CH_2OH$ | 2 O_2 | 1	*over* | *in* $CH_3 - CHO$ | 2 H_2O | 2

Figure 7: Chemical Reactions

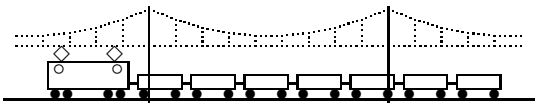


Figure 8: An example Freight Train

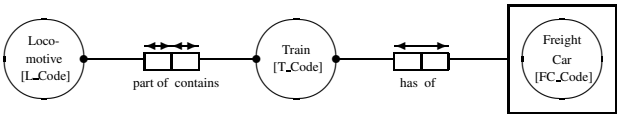


Figure 9: The train composition administration

consider a freight train as depicted in figure 8. A train is identified by a train code, and consists of a locomotive followed by a sequence of freight cars. This Universe of Discourse is modelled in the information structure diagram of figure 9.

5 Relation with Context-Free Grammars

In this section the relation between context-free grammars and the Predicator Set Model is discussed. First we show how a context-free grammar is translated into a Predicator Set schema. In [HW93] a formalised translation mechanism is given.

Context-free grammars are generally employed for describing document structures (see for example [BW90], [SDBW91], [BW92]). The Predicator Set Model has sufficient expressive power to describe such structures elegantly. This is done by interpreting context free grammars in terms of the Predicator Set Model. The translation also shows the usefulness of the Predicator Set Model for describing hypertext information structures. In the translation, generalised object types will play a crucial role. In [HW93] the formalised translation mechanism is given. The reverse process, i.e translating a Predicator Set schema into a context free grammar, is discussed there as well.

In the world of documents a lot of effort has been put into the design of standards for the communication and denotation of document structures and contents ([ISO86]). In this example the following grammar, which is denoted in the style of SGML, is considered for describing the structure of a book.

$$
\begin{aligned}
\langle \text{book} \rangle &\rightarrow \langle \text{title} \rangle \, \langle \text{contents} \rangle \\
\langle \text{contents} \rangle &\rightarrow \langle \text{chapter} \rangle^{+} \\
\langle \text{chapter} \rangle &\rightarrow \langle \text{title} \rangle \, \langle \text{sections} \rangle \\
\langle \text{sections} \rangle &\rightarrow \langle \text{section} \rangle^{+} \\
\langle \text{section} \rangle &\rightarrow \langle \text{string} \rangle \\
\langle \text{title} \rangle &\rightarrow \langle \text{string} \rangle \\
\langle \text{string} \rangle &\rightarrow \langle \text{char} \rangle^{+}
\end{aligned}
$$

This grammar can be translated to the Predicator Set Model schema of figure 10. In this figure, the predicators of fact types have been drawn seperately. The translation is directly derived from the grammar rules: each nonterminal symbol becomes an entity type, and each terminal symbol a label type. Each production rule describes a specifier of the object type corresponding to the lefthand side.

It is important to note that the Predicator Set schema resulting from the translation of a context-free grammar does not exhibit explicitly the order of the symbols in the righthand side of production rules. This corresponds to a *mapping oriented* view to the righthand side of a production rule, rather than the usual tuple oriented view. The resulting Predicator Set schema can be viewed as a representation of the abstract syntax ([Mey90]) corresponding to the grammar at hand.

The grammar box is used as a notation to incorporate in this way context free grammars in the Predicator Set Model. The grammar box takes as inputs the object types that correspond to terminal symbols. The output of the grammar box is the start symbol.

With respect to this use of context-free grammars, a bad schema will result if the context-free grammar does not satisfy some aesthetical rules. Firstly, there can be useless symbols, i.e. symbols that do not occur in any derivation from the start symbol. In terms of the Predicator Set Model these symbols correspond to isolated object types. Secondly, the object types that correspond to the terminal symbols can be identified without making use of the grammar box, since they are interpreted as label types. The identification of the object type, corresponding with the start symbol of the grammar, then depends on the structure of the grammar.

6 Schema Decomposition

In this section a notational shorthand for schema objectification, which facilitates the specification of complex object types is introduced. This is done by discussing some examples, which make use of this short-

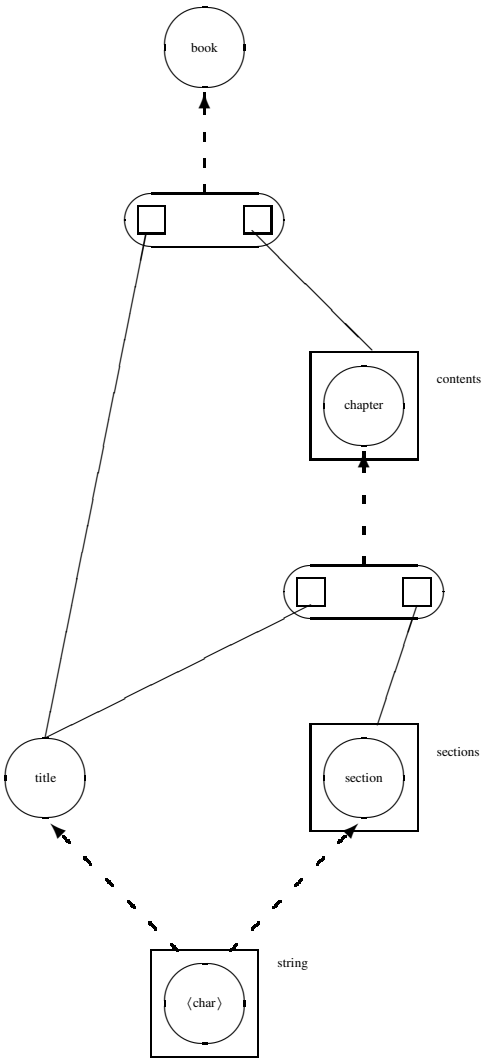


Figure 10: Example of translation of SGML structure

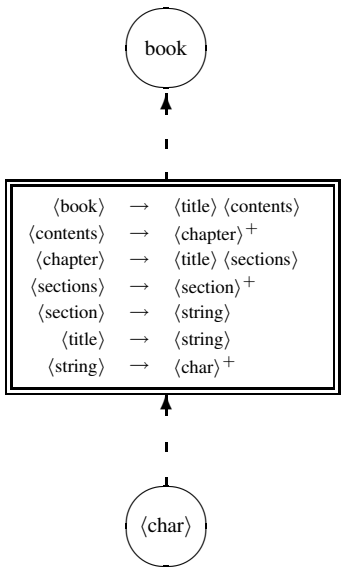


Figure 11: The use of a grammar box

hand, in order to demonstrate its elegance in modelling.

The need for decomposition in large systems has been generally recognised. A well known example is the decomposition mechanism for Activity Graphs ([Sch84]). In an Activity Graph, both processes and data may be subject to decomposition. However, data modelling techniques usually do not provide a decomposition mechanism. In the Predicator Set Model, schema objectification has been introduced for this purpose.

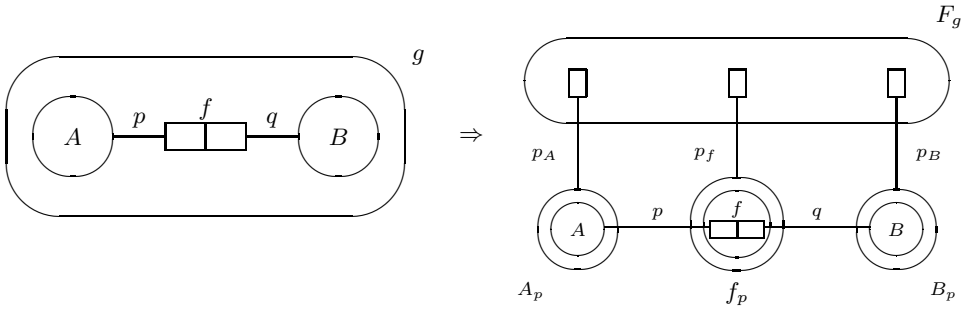


Figure 12: Schema objectification

Schema objectification is a construction mechanism that allows us to define part of a schema as an object type. Instances of such object types are then populations of their corresponding schemas. As a result, these objectified schemas have to be valid information structures, i.e. Predicator Set Model. Furthermore, populations should satisfy the *decomposition rule*, meaning, that instances of an object type O should be valid populations of the objectified schema as well.

Schema objectification, however, is not an elementary concept, since it can be defined in terms of the concepts of power object type and fact type. The idea is to construct a power object type x_p for each object type x from the schema g to be objectified. Each of these power object types x_p is the base of a predicator p_x, that is part of a fact type F_g (see figure 12). This fact type is to relate sets of instances of the object types involved in the schema objectification, which are part of th same schema instance.

As an example of schema objectification, consider a meta-model for Activity Graphs. In figure 13 two sample activity graphs are depicted.

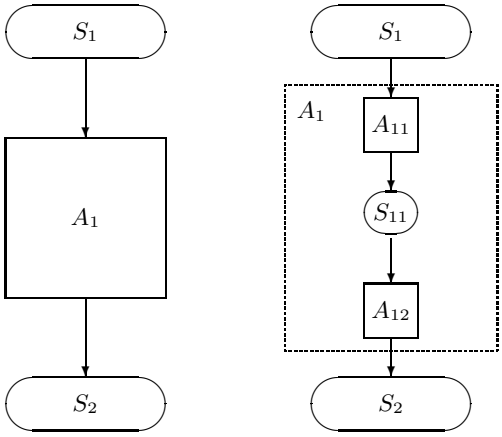


Figure 13: Sample activity graphs

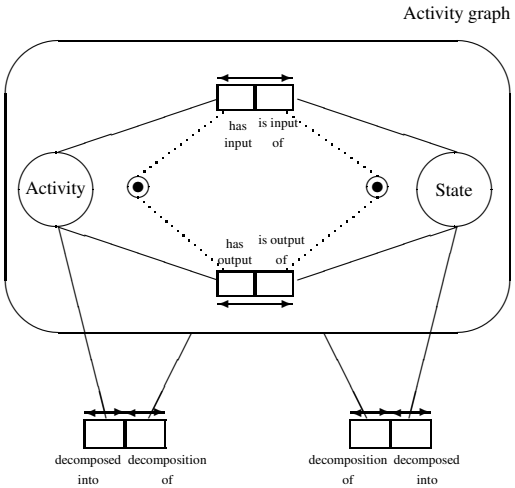


Figure 14: Meta-model of activity graphs

Activity Graphs are bipartite directed graphs consisting of activities and states. The direction of the arrow between an activity and a state indicates whether that state is input or output of that activity. Activities and states can be decomposed in other Activity Graphs. In figure 13 the rightmost Activity Graph shows the decomposition of activity A1.

In figure 14, the meta-model of Activity Graphs is depicted. As can be seen there, an Activity Graph is an objectified schema consisting of activities, states and input and output relations. The binary relations between activity and Activity Graph and state and Activity Graph represent the decomposition relation.

7 Conclusions

The suitability of the Predicator Set Model for complex application domains has been illustrated by means of a number of examples. The theoretic background has been described in [HW93]. In that paper, the relation with contex free grammars and formal set theory has been established, thus giving evidence, from a formal point of view, for the completeness of the Predicator Set Model.

In the future a method will be developed to support the construction of schemata from informal descriptions. Heuristics and guidelines should be ingredients of this methods. A prototype implementation is considered.

References

[Ari91] G. Ariav. Temporally oriented data definitions: Managing schema evolution in temporally oriented databases. *Data & Knowledge Engineering*, 6(6):451–467, 1991.

[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.

[BW90] P. D. Bruza and Th. P. van der Weide. Two level hypermedia - an improved architecture for hypertext. In A.M. Tjoa and R. Wagner, editors, *Proceedings of the Data Base and Expert System Applications Conference (DEXA 90)*, pages 76–83, Vienna, Austria, 1990. Springer-Verlag.

[BW92] P.D. Bruza and Th.P. van der Weide. Stratified Hypermedia Structures for Information Disclosure. *The Computer Journal*, 35(3):208–220, 1992.

[Che76] P.P. Chen. The entity-relationship model: Towards a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, March 1976.

[DMV88] O.M.F. De Troyer, R. Meersman, and P. Verlinden. RIDL* on the CRIS Case: A Workbench for NIAM. In T.W. Olle, A.A. Verrijn-Stuart, and L. Bhabuta, editors, *Information Systems Design Methodologies: Computerized Assistance during the Information Systems Life Cycle*, pages 375–459, Amsterdam, The Netherlands, EU, 1988. North-Holland/IFIP WG8.1.

[Fal93] E.D. Falkenberg. An Approach to Deterministic Event-Tuned Information Analysis. In G.M. Nijssen, editor, *Proceedings of NIAM-ISDM*. NIAM-GUIDE, September 1993.

[FOP92] E.D. Falkenberg, J.L.H. Oei, and H.A. Proper. A Conceptual Framework for Evolving Information Systems. In H.G. Sol and R.L. Crosslin, editors, *Dynamic Modelling of Information Systems II*, pages 353–375. North-Holland, Amsterdam, The Netherlands, EU, 1992. ISBN 0444894055

[HK87] R. Hull and R. King. Semantic Database Modelling: Survey, Applications and Research Issues. *ACM Computing Surveys*, 19(3):201–260, September 1987.

[HPW93] A.H.M. ter Hofstede, H.A. 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.

[HW92] A.H.M. ter Hofstede and Th.P. van der Weide. Formalisation of techniques: chopping down the methodology jungle. *Information and Software Technology*, 34(1):57–65, January 1992.

[HW93] A.H.M. ter Hofstede and Th.P. van der Weide. Expressiveness in conceptual data modelling. *Data & Knowledge Engineering*, 10(1):65–100, February 1993.

[ISO86] *Information Processing – Text and Office Systems – Standard General MarkUp Language (SGML)*, 1986. ISO 8879:1986.
`http://www.iso.org`

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

[Mey90] B. Meyer. *Introduction to the Theory of Programming Languages*. Prentice-Hall, Englewood Cliffs, New Jersey, 1990.

[MS90] E. McKenzie and R. Snodgrass. Schema evolution and the relational algebra. *Information Systems*, 15(2):207–232, 1990.

[NH89] G.M. Nijssen and T.A. Halpin. *Conceptual Schema and Relational Database Design: a fact oriented approach*. Prentice-Hall, Sydney, Australia, 1989. ASIN 0131672630

[Rod91] J.F. Roddick. Dynamically changing schemas within database models. *The Australian Computer Journal*, 23(3):105–109, August 1991.

[Sch84] G. Scheschonk. *Eine auf Petri-Netzen basier-en-de Konstruk-tion-s, Ana-ly-se und (Teil)Verifica-tion-s-me-tho-de zur Modellierungsunterstützung bei der Entwicklung von Informationssystemen*. PhD thesis, Berlin University of Technology, Berlin, Germany, 1984. (In German).

[SDBW91] P.L. van der Spiegel, J.T.W. Driessen, P.D. Bruza, and Th.P. van der Weide. A Transaction Model for Hypertext. In D. Karagiannis, editor, *Proceedings of the Data Base and Expert System Applications Conference (DEXA 91)*, pages 281–286, Berlin, Germany, 1991. Springer-Verlag.

[SS86] H.J. Schek and M.H. Scholl. The relational model with relation-valued attributes. *Information Systems*, 11(2):137–147, 1986.

[VHW91] T.F. Verhoef, A.H.M. ter Hofstede, and G.M. Wijers. Structuring modelling knowledge for CASE shells. In R. Andersen, J.A. Bubenko, and A. Sølvberg, editors, *Proceedings of the Third International Conference CAiSE'91 on Advanced Information Systems Engineering*, volume 498 of *Lecture Notes in Computer Science*, pages 502–524, Trondheim, Norway, May 1991. Springer-Verlag.

[Wel88] R.J. Welke. The CASE Repository: More than another database application. In W.W. Cotterman and J.A. Senn, editors, *Proceedings of 1988 INTEC Symposium Systems Analysis and Design: A Research Strategy*, Atlanta, Georgia, 1988. Georgia State University.

[WHB92] Th.P. van der Weide, A.H.M. ter Hofstede, and P. van Bommel. Uniquest: Determining the Semantics of Complex Uniqueness Constraints. *The Computer Journal*, 35(2):148–156, April 1992.

[Wig90] R.E. Wiggins. Document Image Processing - New Light on an Old Problem. *International Journal of Information Management*, 10(4):297–318, 1990.

[Win90] J.J.V.R. Wintraecken. *The NIAM Information Analysis Method: Theory and Practice*. Kluwer, Deventer, The Netherlands, EU, 1990.