Mapping from a Conceptual Schema to a Target Internal Schema

D. J. FLYNN* AND A. H. F. LAENDER†
School of Computing Studies and Accountancy, University of East Anglia, Norwich NR4 7TJ

This paper describes the mapping from a NIAM schema into the ACS, a formalism which has been advanced as a candidate for an internal representation of a conceptual schema. Aspects of NIAM and ACS are described briefly, and an application of the mapping algorithm between NIAM and ACS is illustrated with an example.

1. INTRODUCTION

1.1. Overview

The ANSI/SPARC three-schema framework has been widely accepted within the database community. However, there still exists some controversy concerning the nature of the conceptual schema. It has been argued[13, 20, 21] that the conceptual schema has a dual role, one being concerned with the modelling of reality, the other concentrating on the logical requirements to be imposed upon the internal schema.

The ACS[19, 21] has been proposed as a candidate for the second of these roles. It is intended to be structural rather than semantic, acting as a suitable target for mappings from a semantic world, but also being of a structural form which may easily be used for computer-implemented query, update, etc. It has two main purposes:

(*a*) to act as the schema which is an intermediate stage in an automatic mapping from conceptual view to implemented system;

(*b*) to act as a mapping target from different schemata (expressed possibly in different languages) with the intention of determining the equivalence of those schemata.

It is the second purpose of the ACS which is described in this paper.

Many database implementations, of different types, may exist within an organisation, and it is felt that, if it is required to integrate these different databases, then the ACS may be a useful tool for the determination of their equivalences, a necessary preliminary to integration. The ACS captures the functionalities and constraints which have been determined by means of a conceptual schema. It is represented as a set of relations which contain this information.[21] Information concerning the real-world model, for example that it was based on some ERA scheme, is lost. The ACS has a set of semantic rules which govern its structure. It is claimed that a real-world situation which can be represented in the ACS leads to a unique representation in these relations,[22] irrespective of the form of the conceptual schema which was used to collect the functionalities and constraints. Use of the ACS as a mapping target for the relevant schemata, and the comparison of the resulting ACSs, together comprise a useful tool for determining equivalence.

* Present address: Department of Computation, University of Manchester Institute of Science and Technology, P.O. Box 88, Manchester, M60 1QD.

† Present address: Departamento de Ciencia da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, Brasil.

Another similar application of the ACS is that of a global schema in a distributed database environment, where it may facilitate mapping between different schemata, possibly expressed in different data models, and possibly containing overlapping data.

1.1.1. *Aim of paper*

The aim of this paper is to describe the problem of mapping from a schema expressed in a commonly used data model, NIAM,[24] into the ACS. The aspect of the mapping that has been considered is only one-way – i.e. from NIAM to ACS. An example of the usefulness of this approach would be the determination of equivalences of two overlapping schemata expressed in NIAM. It is a generally accepted view that such mappings are relatively straightforward, although little has been published in this area.[3, 4, 15] However, it may be noted that data models are increasing in complexity, being extended to model a wider range of semantic constraints,[24] of transactions,[5] and of time.[11] Mappings between these models may be neither simple, nor possible, in some cases.

The significant feature of the ACS which makes it suitable as such a tool is its intention to be free of isomorphisms. By this is meant that the mapping from any given schema into the ACS can be expressed in only one way. Expressed differently, the results of mapping the same schema into ACS by different persons should be identical. This avoids the use of complex checking procedures within ACS to determine equivalences. The current state of the ACS is that of working towards removing isomorphisms, whilst at the same time taking care to preserve the human intelligibility needed for the comparison process. However, it is assumed that the humans possess some expert computer knowledge.

1.2. Description of contents

The main part of the paper is concerned with a mapping from an NIAM schema, giving the ACS result in graphical form. The NIAM schema is taken from the 'base' NIAM paper,[24] which includes a schema as a partial solution to the recent IFIP Conference problem.[18] A more detailed version is available in Ref. 8.

The nature of the source and target for the mapping may be briefly mentioned here. The graphical modes of both NIAM and ACS were unsuitable for these roles, as they were incomplete, especially with regard to constraints. It was felt necessary to simplify the mapping as much as possible, and so certain assumptions about the nature of

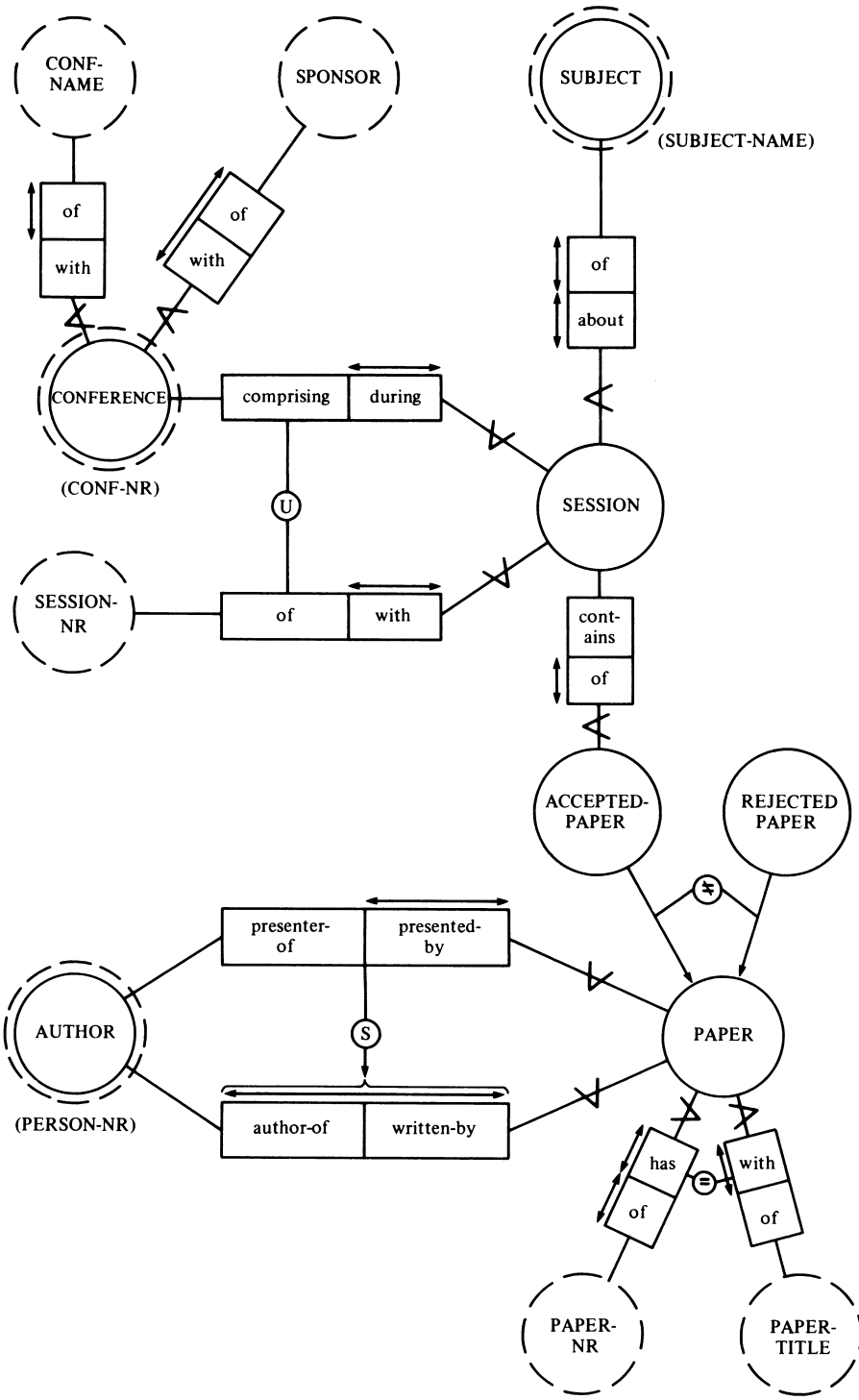


Figure 2.1. NIAM source ISD.

the NIAM source and the ACS target were made. For NIAM, it was assumed that the constructs could be allocated to pre-defined sets. For ACS, as the mode of representation was not decided at the time of this work, sets which were similar to the source sets were used as the mapping target. As the results of the mapping had to be demonstrated, the decision was made to use the graphical mode of NIAM and ACS as the vehicle for illustrating the applicability of the mapping algorithm. This would also make examples easier to follow.

In Sections 2 and 3, the basic constructs of NIAM and ACS are described, with examples. The principles of mapping between the constructs are outlined in Section 4, and an application of the mapping algorithm to a NIAM example is presented in Section 5. Section 6 is concerned with mapping extensions, and conclusions are drawn in Section 7.

2. NIAM – INTRODUCTORY DESCRIPTION

2.1. Introduction

In this section an introduction to the concepts and constructs of NIAM is given.

NIAM is supported by a meta-information system called ISDIS, a computer-assisted system which helps to maintain all the results produced during the development of an information system. These results are expressed in a formal language RIDL, which is the communication language between ISDIS and its users.

The two interrelated areas of data and process analysis are addressed in NIAM. Functional decompositions and Information Flow Diagrams are produced, and Information Structure Diagrams (ISD), which are data structure and constraint diagrams, are then designed. The ISD is only intended to be a graphical tool, and RIDL allows one to express features other than those in the ISD. For example, transactions are not modelled in the ISD, but are expressed in RIDL. As our interest in this paper is with mappings from a conceptual schema into the ACS, the majority of the NIAM features which are relevant to this are in the ISD.

2.2 ISD constructs

NIAM[24] is based on a binary model of data, and has grown out of the work of Falkenberg[6,7] and Nijssen.[16,17] It is also strongly featured in the recent ISO report.[14] A NIAM ISD is shown in Fig. 2.1.

2.2.1. *Basic constructs*

NIAM considers object types and relationships, shown respectively as circles and edges in the graphical form. Object types are either NOLOTs or LOTs, shown by solid and broken circles respectively. A NOLOT (nonlexical object type) is an object of which occurrences cannot be shown. It is an example of the concept known as the 'carrier' object. Examples are PAPER and CONFERENCE. A LOT (lexical object type) is an object, occurrences of which may be shown. Typically, it is a string or an integer. For example: '123' is an occurrence of the LOT PERSON-NR. NOLOT occurrences and LOT occurrences are 'things' and 'properties of things' respectively.

2.2.2 *Associations*

Associations are of two types, bridge types and idea types. A bridge type is an association between a NOLOT and a LOT. For example, the NOLOT PAPER is associated to the LOT PAPER-TITLE. The word 'bridge' is significant, as a bridge type is often used to identify a NOLOT. For example, CONFERENCE is associated with the LOT CONF-NR. (The graphical form for this in Fig. 2.1 is a shorthand.) An idea type associates two NOLOTs – for example, PAPER and AUTHOR. NIAM captures the semantics of associations between two objects with the Role name. For example, AUTHOR has a role PRESENTER-OF with PAPER; similarly, PAPER has a role PRESENTED-BY with AUTHOR.

2.2.3. *Subtypes*

The subtyping feature in NIAM is identical to the commonly accepted notion of an object in a generalisation hierarchy inheriting properties from its generic while possessing certain distinguishing associations of its own. However, subtypes may not necessarily be disjoint and may overlap. Fig. 2.1 illustrates two subtypes of PAPER – ACCEPTED-PAPER and REJECTED-PAPER. Arrowed edges indicate subtype and generic, the arrow pointing towards the generic.

2.2.4. *Constraints*

NIAM models six types of constraints in the ISD. The definition of constraint to be found in Ref. 24 is as follows: 'any abstraction system will include rules which prescribe the behaviour of the object system – these rules are called constraints'. The constraint types are as follows.

IDENTIFIER CONSTRAINT. This constrains the degree of an association to be 1:1, 1:N or M:N, and is shown by arrows above the role names of the association. For example, the association between SESSION and SUBJECT is 1:1. SESSION may be ABOUT only one SUBJECT, and a SUBJECT may be OF only one SESSION. A SESSION may be DURING one CONFERENCE, while a CONFERENCE is COMPRISING many SESSIONs (1:N, N:1). Finally, a PAPER may be WRITTEN-BY many AUTHORs, and an AUTHOR may be AUTHOR-OF many PAPERs (M:N). The constraint may apply to both bridge types and idea types.

SUBSET CONSTRAINT. This constraint is on the joint population of two roles played by the same two objects, and is indicated by an 'S' in a circle. In the figure, the constraint is that the PRESENTER of a PAPER must be one of its AUTHORs.

EQUALITY CONSTRAINT. This constrains the population of a role (i.e. the number of objects of a certain object type playing that role) to be equal to the population of another role for the same object type, and is shown by an '=' in a circle. The figure shows how a PAPER with a PAPER-NR must also have a PAPER-TITLE.

UNIQUENESS CONSTRAINT. This constraint specifies that a NOLOT occurrence is uniquely identified by a combination of role occurrences from different bridge or idea types. In a sense this is not a constraint at all, but simply a means of defining the source of identifiers for objects (within the confines of the binary model) which require a compound identifier – i.e. which involve more than one association. It is shown by a 'U' in a circle. The figure shows how the SESSION NOLOT is identified by occurrences of CONFERENCE and SESSION–NR.

DISJOINT CONSTRAINT. Objects of a certain type are constrained to be disjoint from objects of another type. For example, the figure shows that a PAPER which is an ACCEPTED-PAPER may not also be a REJECTED-PAPER – this is indicated by a '$\neq$' in a circle.

TOTAL ROLE CONSTRAINT. The meaning of this constraint is that all the objects of an object type must be associated with another object type – i.e. they must all play a certain role. For example, the constraint that all ACCEPTED-PAPERs are to have an association with a SESSION is shown by a 'V' across the relevant edge.

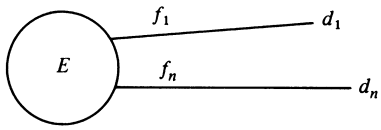
3. THE ABSTRACTED CONCEPTUAL SCHEMA (ACS)

The basic construct in the ACS is the element. The element exists only in concept and its purpose is to carry what one might call indirect data functions.[21] Associated with an element are a number of properties, each property

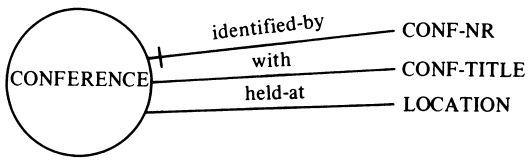
corresponding to a function which maps the element on to a specific data item value. Thus the data item is the basic unit of recorded data. An element set is a collection of all elements which have associated with them the same set of properties (function/data item combinations). Furthermore, an element must be associated with all of its properties – i.e. no partial function is allowed.

A particular property or set of properties which associates its data item values with one and only one element of a set is said to be the element set identifier. Names are associated with data items, functions and element sets, and as a property may be associated only once with an element the function/data item names must be unique throughout the database.

A schematic may be seen below. *E* is the name of the element set and the *f*s and *d*s are the names of the functions and data items respectively. The identifier is shown by a stroke through one or more function/data item edges.



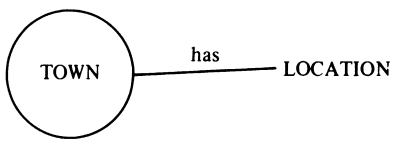
An element set is clearly the target for representing an 'entity class'. So, in the example described below, the element set representing a possible 'entity class' CONFERENCE is given. The property 'identified by/CONF-NR' is the element set identifier.



In the ACS, integrity constraints are specified by means of value sets. A value set is defined by specifying an element set and one or more properties, and is the set of (tuples of) data item values taken as the properties are applied to each element of the set. To define a value set constraint we write a statement of the form:

V1 ⟨operator⟩ V2,

where V1 and V2 are pre-defined value sets. The basic operators used are the set-oriented operators . is-equal-to . , . is-disjoint-to . , . is-contained-in . ,and . contains-all. Value sets may be operated on to derive new ones, so that complex constraints may be expressed. For example, in Fig. 3.1 the value set constraint V1 . is-contained-in . V2 specifies that a conference location must be in an already existing set of towns.



V1 = CONFERENCE (held-at/LOCATION)
V2 = TOWN (has/LOCATION)

V1 . is-contained-in . V2

Figure 3.1. Example of value-set constraint.

To capture the notion of 'role'[2] and 'generalisation' in the semantic data models,[23] an element set may be defined as located within another one, so inheriting all its properties. For example, the element set IFIP-CONFERENCE is located in the element set CONFERENCE. This is shown graphically in Fig. 3.2, where the arrowed edge indicates the locating relationship.

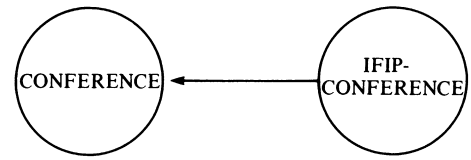


Figure 3.2. Graphical subtype representation in ACS.

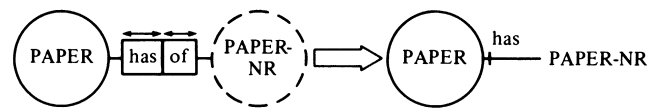
The ACS constructs defined above are those which correspond to the initial proposal described in Ref. 21 and are used in this work. However, a number of extensions have been proposed recently and the interested reader is referred to this.[19]

4. NIAM – ACS MAPPING OUTLINE

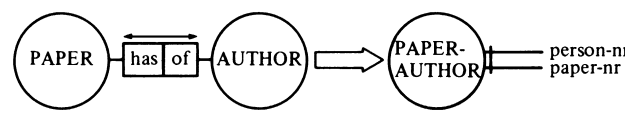
A NIAM schema contains the basic constructs NOLOT and LOT, which map to element set and data item respectively. Associations between constructs, which are modelled by bridge types and idea types, are mapped to ACS in a way which depends chiefly on the functionality of the association. However, the semantics of the role names involved are mapped directly to function names. An inverse function name must also be present. Two important rules guide the mapping.

First, all ACS functions are total. This requirement has the consequence that a NOLOT may map to more than one element set, if it has functional associations with objects which are not subject to the Total Role Constraint (see section 2.2.4).

Secondly, the associations between element sets are represented by their identifiers. When an NIAM association is functional – i.e. a Type 1 or Type 2 bridge type or idea type, then the relevant data item (which may be an element set identifier) is (functionally) associated with an ACS element set by the formation of a property – for example:



When the association is not functional, (1:*N*, *M*:*N*), then for *M*:*N* a 'carrier' element set is created with a (compound) identifier drawn from the identifiers of the two objects in the *M*:*N* association – for example:



PERSON-NR and PAPER-NR are the identifiers of PAPER and AUTHOR respectively. The 1:*N* association is treated simply as the inverse of an *N*:1 functional association.

Constraints map, mostly, to value-set constraints. However, some constraints in NIAM are structurally

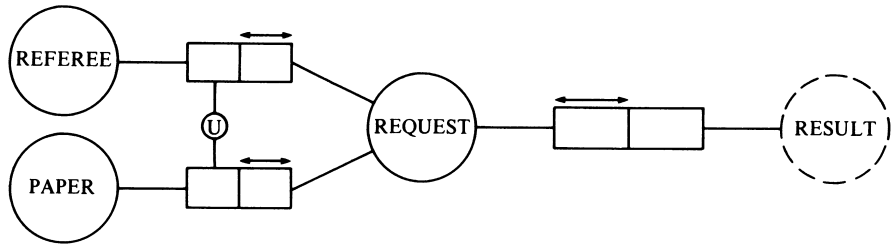


Figure 4.1. Example of a nested binary association in NIAM.

embedded in ACS. An example is the Identifier Constraint. Similarly, the Equality and Total Role Constraints are implicit in the total function feature of ACS. The disjoint constraint, however, requires a set expression and thus the value-set constraint is used.

It should be noted that although NIAM basically uses binary associations for modelling reality, *n*-ary and nested binary associations[6] may also be captured. This is done by the creation of an 'artificial' NOLOT, identified by the relevant roles using the Uniqueness Constraint. This does not affect the mapping to ACS. For example, Fig. 4.1 shows a nested binary association whereby REQUEST is associated with RESULT, and REQUEST is defined by a Uniqueness Constraint on the NOLOTs REFEREE and PAPER.

The well-known Supplier, Part and Project problem is an example of an *n*-ary association. In this case, as the Uniqueness Constraint would involve all three components, the identifier of the corresponding ACS element set would involve the identifiers of the three defining element sets.

5. MAPPING ALGORITHM

The formal rules of the Mapping Algorithm are available elsewhere.[8] Here an informal account is presented of an application of the algorithm to the NIAM ISD in Fig. 2.1. Intermediate results of the different stages of the algorithm are presented. The final ACS mapping is presented in Figs. 5.2 and 5.3.

5.1. Assign the intermediate state sets for NIAM constructs

The sets are in Fig. 5.1. S3 contains NOLOTs formed by Uniqueness Constraints; sets S5–S8 contain, respectively, bridge types 1–4, and sets S9, S10 and S11 contain idea types 1, 2 and 4.

5.2. LOTS and NOLOTS

5.2.1. *Map all LOTs to data items*
5.2.2. *Map all NOLOTs, not subtypes or those defined by uniqueness constraints, to element sets*

These operations use the sets S1 and S2 to produce the element sets AUTHOR, CONFERENCE, SUBJECT and PAPER, and the data items PERSON-NR, CONF-NR, SESSION-NR, SUBJECT-NAME, PAPER-NR, PAPER-TITLE, CONF-NAME and SPONSOR.

5.3 Type 1 bridge types

This step assigns identifiers to the element sets created. *From set S5, construct a property consisting of a data item, function name and inverse function name corresponding to the LOT and the role names R1 and R2, and associate this property with the element set corresponding to the NOLOT. Designate this property as the identifier.*

This operation, and subsequent ones, produce element sets and properties which will be expressed here according

*S*1 ⟨*LOT*⟩
Person-nr Conf-nr Session-nr Paper-nr Paper-title
Sponsor Conf-name Subject-name
S3 ⟨*NOLOT*⟩⟨$NOLOT_1$⟩⟨LOT_1⟩⟨$R1_1$⟩⟨$R2_1$⟩
. . . ⟨$NOLOT_n$⟩⟨LOT_n⟩⟨$R1_n$⟩⟨$R2_n$⟩
Session, Conference, during, comprising, session-nr,
with, of
*S*4 ⟨*GENERIC*⟩⟨*SUBTYPE*⟩
Paper, Accepted-paper
Paper, Rejected-paper

*S*6 ⟨*NOLOT*⟩⟨*LOT*⟩⟨*R*1⟩⟨*R*2⟩
Session, session-nr, with, of
Paper, paper-title, with, of
*S*8 ⟨*NOLOT*⟩⟨*LOT*⟩⟨*R*1⟩⟨*R*2⟩
Conference, sponsor, with, of
*S*10 ⟨*NOLOT*1⟩⟨*NOLOT*2⟩⟨*R*1⟩⟨*R*2⟩
Session, conference, during, comprising
Paper, author, presented-by, presenter-of
Accepted-paper, session, of, contains

*S*2 ⟨*NOLOT*⟩
Author Conference Subject Paper

*S*5 ⟨*NOLOT*⟩⟨*LOT*⟩⟨*R*1⟩⟨*R*2⟩
Conference, conf-nr, has, of
Paper, paper-nr, has, of
Author, person-nr, has, of
Subject, subject-name, has, of
*S*7 ⟨*NOLOT*⟩⟨*LOT*⟩⟨*R*1⟩⟨*R*2⟩
Conference, conf-name, with, of

*S*9 ⟨*NOLOT*1⟩⟨*NOLOT*2⟩⟨*R*1⟩⟨*R*2⟩
Session, subject, about, of
*S*11 ⟨*NOLOT*1⟩⟨*NOLOT*2⟩⟨*R*1⟩⟨*R*2⟩
Author, paper, author-of, written-by

Figure 5.1. NIAM source sets for mapping algorithm.

No.	ELEMENT-SET/PROPERTY	OPERATOR	ELEMENT-SET/PROPERTY
V1	ACCEPTED-PAPER(has/PAPER-NR)	ce	PAPER(has/PAPER-NR)
V2	REJECTED-PAPER(has/PAPER-NR)	ce	PAPER(has/PAPER-NR)
V3	SESSION(during/CONF-NR)	ce	CONFERENCE(has/CONF-NR)
V4	CONF-NAME(of/CONF-NR)	ce	CONFERENCE(has/CONF-NR)
V5	CONF-SPONSOR(of/CONF-NR)	ce	CONFERENCE(has/CONF-NR)
V6	SUBJECT(of/CONF-NR, of/SESSION-NR)	ce	SESSION(during/CONF-NR, with/SESSION-NR)
V7	SESSION(about/SUBJECT-NAME)	ce	SUBJECT(has/SUBJECT-NAME)
V8	ACCEPTED-PAPER(of/CONF-NR, of/SESSION-NR)	ce	SESSION(during/CONF-NR, with/SESSION-NR)
V9	PAPER(presented-by/PERSON-NR)	ce	AUTHOR(has/PERSON-NR)
V10	AUTHOR-PAPER(written-by/PERSON-NR)	ce	AUTHOR(has/PERSON-NR)
V11	AUTHOR-PAPER(author-of/PAPER-NR)	ce	PAPER(has/PAPER-NR)
V12	PAPER(has/PAPER-NR, presented-by/PERSON-NR)	ce	AUTHOR-PAPER(author-of/PAPER-NR, written-by/PERSON-NR)
V13	ACCEPTED-PAPER(has/PAPER-NR)	cn	REJECTED-PAPER(has/PAPER-NR)

Note: ce = ● is-contained-in ●; cn = ● is-disjoint-of ●.

Figure 5.2. ACS Value sets.

to this template: ⟨ELEMENT SET⟩ ⟨FUNCTION NAME⟩ ⟨INVERSE FUNCTION NAME⟩ ⟨DATA ITEM⟩. In this case the operation produces these associations:

AUTHOR, has, of, PERSON-NR
CONFERENCE, has, of, CONF-NR
SUBJECT, has, of, SUBJECT-NAME
PAPER, has, of, PAPER-NR.

5.4 Subtypes

From set S4, create an element set for the subtype; obtain the constituent properties of the identifier of the generic, and associate these properties with the element set corresponding to the subtype. Designate these properties as the identifier, and locate the subtype element set in the generic element set. Define a value set constraint on these properties.

This operation produces:

ACCEPTED-PAPER, has, of, PAPER-NR
REJECTED-PAPER, has, of, PAPER-NR
PAPER: ACCEPTED-PAPER and PAPER: REJECTED-PAPER subtypes.

Value-set constraints V1 and V2 (see Fig. 5.2 for value-set constraints).

5.5. Type 2 bridge types

Using set S6, construct a property consisting of a data item, function name and inverse function name corresponding to the LOT, R1 and R2, and associate this property with the element set corresponding to the NOLOT. However, where a Total Role Constraint does not exist – i.e. where not all NOLOT occurrences are in association with this LOT, the element set corresponding to the NOLOT cannot be associated with the property, as ACS does not allow partial functions. Instead, an element set must be generated which is a subtype of the element set corresponding to the NOLOT, and the property is associated with this. Only NOLOTs in set S2 are considered, hence the operation produces: PAPER, with, of, PAPER-TITLE.

5.6. NOLOTS defined by uniqueness constraints

All remaining NOLOTs (those defined by uniqueness constraints, and their subtypes) are now mapped. Identifiers, type 1 and type 2 bridge types are associated. Subtypes are located. The detail is almost identical to steps 5.3, 5.4 and 5.5. The set used is S3.

5.6.1. *Map the NOLOT to an element set*
5.6.2. *Procedure for obtaining the identifier of the element set*

For each LOT defining the NOLOT that is subject to the uniqueness constraint, construct a property from the data item corresponding to the LOT; for each NOLOT defining the NOLOT that is subject to the uniqueness constraint, obtain the constituent properties of the defining NOLOT identifier(s) and construct properties from the data item(s) involved. Use the corresponding role names as function and inverse function names. Associate the properties with the element set and designate as (compound) identifier. Define value-set constraints on these properties.

This operation produces the element set name SESSION, and the associations:

SESSION, during, comprising, CONF-NR
SESSION, with, of, SESSION-NR
(Value-set constraint V3).

5.6.3. *Associate Type 1 bridge types with the element set*

They may already have been used to identify the element set. In this case, none exists.

5.6.4. *Create subtype element sets and locate them within the generic*

Again, none exists.

5.6.5. *Associate Type 2 bridge types with the element set*

In this case the one that exists has been mapped already as part of the identifier.

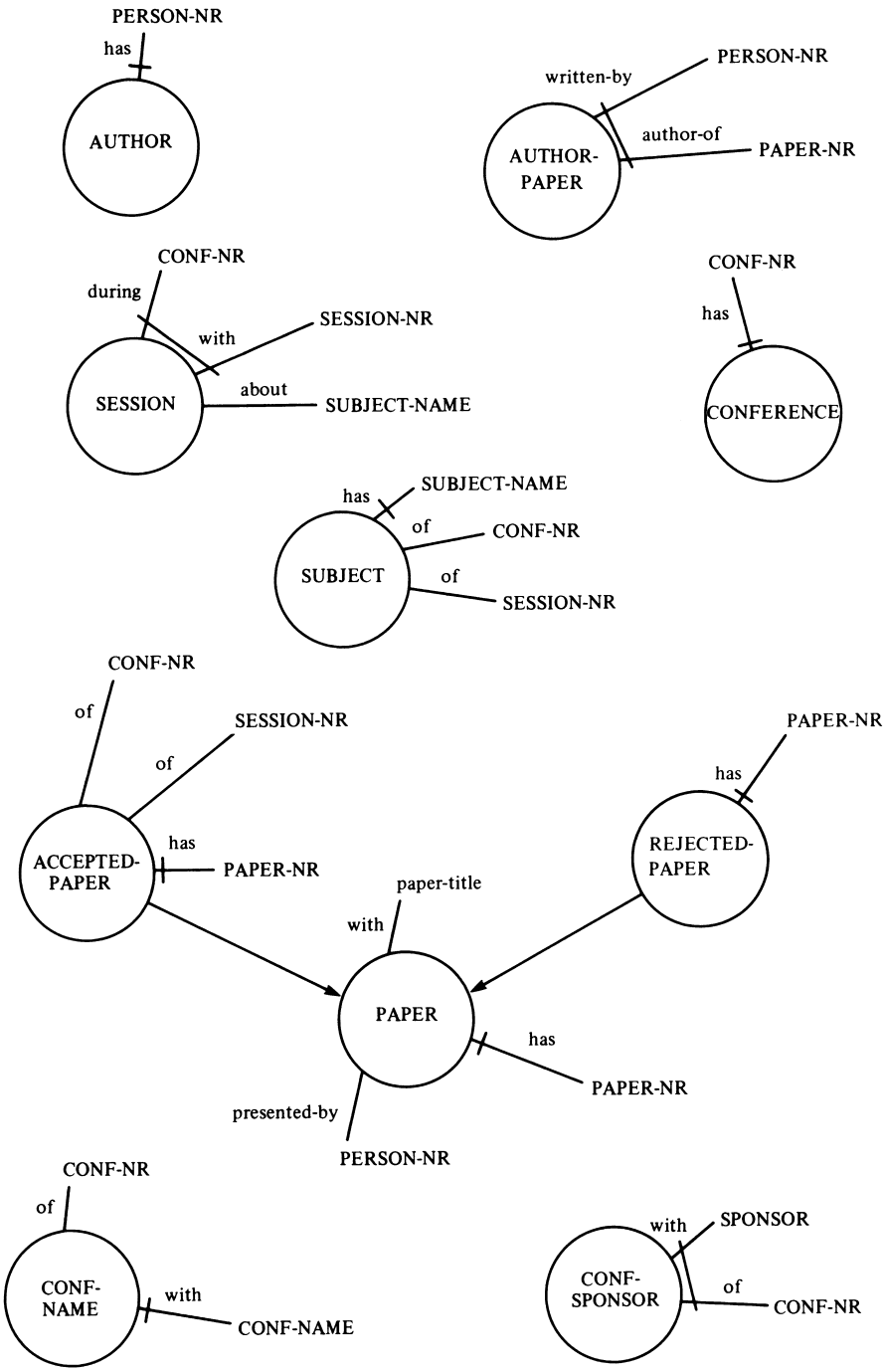


Figure 5.3. Final ACS mapping – graphical constructs.

5.6.6. *Map the next NOLOT, until all are completed*

5.7 Type 3 bridge type

By this stage identifiers have been established for all element sets, including subtypes. This is necessary for the next stages, all of which entail the use of identifiers to model the NIAM associations.

Create an element set using both the LOT and NOLOT names. Construct a property consisting of a data item, function name and inverse function name corresponding to the LOT and the role names R1 and R2, and associate this with the element set created. Designate this property as the identifier. Obtain the constituent properties of the NOLOT identifier and construct properties from the data items involved, using R2 and R1 as function and inverse function names, and associate these properties to the element set created. Define a value-set constraint on these properties.

The operation uses set S7, producing the element set CONF-NAME, with the associations:

CONF-NAME, with, of, CONF-NAME
CONF-NAME, of, has, CONF-NR
(Value-set constraint V4).

5.8. Type 4 bridge type

Create an element set corresponding to the NOLOT-LOT combination, using both their names. Construct a property

consisting of a data item, function name and inverse function name corresponding to the LOT and the role names R1 and R2, and associate this with the element set created. Obtain the constituent properties of the NOLOT identifier and construct properties from the data items involved, using R2 and R1 as function and inverse function names, and associate these properties with the element set created. Define a value set constraint on these properties. Designate all properties associated with the element set as (compound) identifier.

This operation produces the element set CONF-SPONSOR, with associations:

CONF-SPONSOR, of, has, CONF-NR
CONF-SPONSOR, with, of, SPONSOR
(Value-set constraint V5).

5.9. Type 1 idea type

Obtain the constituent properties of the NOLOT1 identifier and construct properties from the data items involved, using R2 and R1 as function and inverse function names, and associate these properties with the element set corresponding to NOLOT2. Define a value set constraint on these properties. Obtain the constituent properties of the NOLOT2 identifier and construct properties from the data items involved, using R1 and R2 as function and inverse function names, and associate these properties with the element set corresponding to NOLOT1. Define a value set constraint on these properties.

The operation produces the value-set constraints V6 and V7 and:

SESSION, about, of, SUBJECT-NAME
SUBJECT, of, about, CONF-NR
SUBJECT, of, about, SESSION-NR.

5.10. Type 2 idea type

Obtain the constituent properties of the NOLOT2 identifier and construct properties from the data items involved, using R1 and R2 as function and inverse function names, and associate these properties with the element set corresponding to NOLOT1. Define a value set constraint on these properties. However, where a Total Role Constraint does not exist, as was seen in 5.5, the element set corresponding to NOLOT1 cannot be associated with the properties, as ACS does not allow partial functions. Instead, an element set must be generated which is a subtype of the element set corresponding to NOLOT1, and the properties associated with this.

This operation produces:

PAPER, presented-by, presenter-of, PERSON-NR
ACCEPTED-PAPER, of, contains, CONF-NR
ACCEPTED-PAPER, of, contains, SESSION-NR
(Value-set constraints V8 and V9).

Other members of set S10 have already been mapped in 5.6.2, as they are used for identifiers.

5.11. Type 4 idea type

Construct an element set corresponding to the NOLOT1-NOLOT2 combination. Obtain the constituent properties of the NOLOT1 identifier and construct properties from the data items involved, using R2 and R1 as function and inverse function names, and associate these properties with the element set created. Define a value set constraint on these properties. Similarly, obtain the constituent properties of the NOLOT2 identifier and construct properties from the data items involved, using R1 and R2 as function and inverse function names, and associate these properties with the element set created. Define a value set constraint on these properties.

The operation produces the element set name AUTHOR-PAPER, and the associations:

AUTHOR-PAPER written-by, author-of, PERSON-NR
AUTHOR-PAPER, author-of, written-by, PAPER-NR
(Value-set constraints V10 and V11).

5.12. Constraints

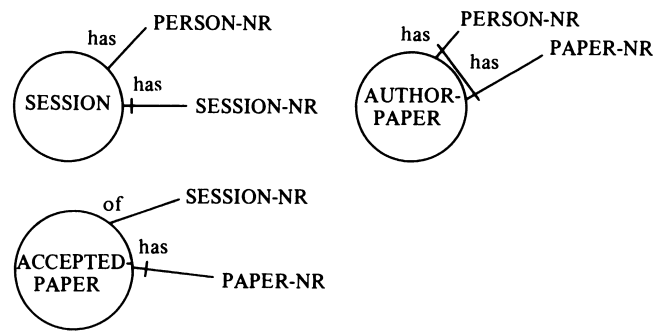
The subset constraint between PAPER and AUTHOR produces value set constraint V12, the equality constraint between PAPER-NR and PAPER-TITLE is met by the ACS total function feature, while the disjoint constraint between ACCEPTED-PAPER and REJECTED-PAPER is mapped into value-set constraint V13.

6. MAPPING EXTENSIONS

Two types of constraints that cannot be expressed in the graphical ISD occur in RIDL. These are: (1) 'non-local' constraints; (2) cardinality constraints.

6.1. Non-local constraints

Some constraints cannot be expressed in the ISD. This can be due to the fact that the graphical objects are some distance apart, making it difficult to achieve a clear graphical representation. However, another reason is that the object being constrained can only be distinguished by a combination of the object name and the role name. This situation occurs when an association is not subject to a total role constraint. Graphical constraints rely on objects being identified by object type or subtype only. Both the subset and the disjoint constraints have been found to occur in situations like this. An example is the constraint that session-chairpersons should be disjoint from authors of papers in their sessions.



V1 = ACCEPTED-PAPER(of/SESSION-NR,has/PAPER-NR)
V2 = AUTHOR-PAPER(has/PERSON-NR,has/PAPER-NR)
V4 = SESSION(has/SESSION-NR,has/PERSON-NR)

V3 = V1 cjoin V2
V4 . is-disjoint-of . V3

Figure 6.1. Derived value set feature.

In ACS, this constraint cannot be modelled at the moment. However, an extension is proposed[19] along the lines of Fig. 6.1, where two or more value-sets may be operated on with relational-like operators. In Fig. 6.1 a 'composition join' operator 'cjoin' is illustrated, for joining value-sets V1 and V2 on the has/PAPER-NR domain, and projecting on this same domain. This feature is termed the *derived value set* feature.

6.2. Cardinality constraint

In RIDL the cardinality of an object type may be restricted. It is proposed that the ACS handle this via the *constant-function* construct of the relevant element set, whereby the element set is associated with a value. This construct is similar to the mechanism in SDM[12] for expressing set properties.

7. CONCLUSIONS

All constraints in the example have been mapped to ACS. The value-set and derived value-set constructs of ACS are able to express most constraints statically, including those in RIDL which are of a procedural form. This is significant, as the holding of constraints in an internal schema at run-time has significant advantages in terms of the integrity of the system, as multiple copies of the same constraint do not have to exist in the application programs accessing that schema. Work is proceeding on this feature of the ACS.[9] The fact that the ACS value-set construct is of such wide applicability as a target for the NIAM constraints should be emphasised, as constraints are usually regarded as the most difficult part of schema mapping.

No major problem was encountered in the mapping, except possibly where the NIAM schema would produce partial functions. As was seen in Section 5.5, ACS does not allow this, and an element set is generated to 'carry' those elements which have the association in question. This may lead to a proliferation of such element sets.

This point leads to the conclusion that a further use for the ACS would be as a design aid to analyse a NIAM schema, as a guide not only to improving the schema, but also as an indicator to an efficient implementation. For example, a large number of partial functions, as mentioned above, might suggest that the NIAM schema would benefit from further use of the subtype feature. Another example is the extra element sets gained by the mapping of Type 3 and Type 4 bridge types, which might call into question the need, as implicitly expressed in the NIAM schema, to store these separately at implementation time.

In addition, it is also felt that a point of theoretical interest is involved, as the ACS may be used to compare various features of different data models. For example, their expressive power, or their duplication of constructs. Exercises comparing the features of three data models,[1] and constraints in four data models,[10] using the ACS, have already been carried out.

The main conclusion reached from this example is that the ACS has, by its capacity to serve as a mapping target from an NIAM schema, proved its capability as a candidate both as a design aid and as a component in an implementation in a 4-level architecture.

Acknowledgements

The authors would like to thank their supervisor Professor P. M. Stocker and their referee for many valuable comments on an earlier version of this paper. Financial support from the S.E.R.C. and the Brazilian Ministry of Education agency C.A.P.E.S. is also gratefully acknowledged.

REFERENCES

1. Charles O. Akinyokun *et al.*, Untitled paper, School of Computing Studies and Accountancy, University of East Anglia (Dec. 1982).
2. C. W. Bachman and M. Daya, The role concept in data models. *Proc. 3rd VLDB Conference, Tokyo, Japan* (Sept. 1977).
3. H. Biller, On the equivalence of data base schemas – a semantic approach to data translation. *Information Systems* **4** (1) (1979).
4. S. A. Borkin, Data model equivalence. *Proc. 4th VLDB Conference, Berlin, Germany* (1978).
5. M. L. Brodie and E. Silva, Active and Passive Component Modelling: ACM/PCM. In Ref. 18.
6. E. Falkenberg, Concepts for modelling information. In *Modelling in Data Base Management Systems*, edited G. M. Nijssen. North-Holland, Amsterdam (1976).
7. E. Falkenberg, Concepts for the coexistence approach to data base management. In *International Computing Symposium 1978*, North-Holland, Amsterdam (1978).
8. D. J. Flynn and A. H. F. Laender, NIAM-ACS mapping rules. Report number CSA/9/1984, School of Computing Studies and Accountancy, University of East Anglia (March 1984).
9. D. J. Flynn, A classification of constraints. Ph.D. thesis (in preparation), School of Computing Studies and Accountancy, University of East Anglia (1985).
10. D. J. Flynn, A comparison of four data models with respect to selected integrity constraints. *Proc. 3rd British National Conference on Databases (BNCOD-3), Leeds, UK, July 1984* forthcoming.
11. M. R. Gustafsson, T. Karlsson, J. A. Bubenko jr. A declarative approach to Conceptual Information Modelling. In Ref. 18.
12. M. Hammer and D. McLeod. Database description with SDM: a semantic data model. *ACM TODS* **6** (3) (Sept. 1981).
13. W. Kent, Splitting the conceptual schema. *Proc. 6th VLDB Conference, Montreal, Canada* (Oct. 1980).
14. ISO/TC97/SC5/WG3. *Concepts and Terminology for the Conceptual Schema and the Information Base*. ISO (1982).
15. A. Klug, Calculating constraints on relational expressions. *ACM TODS* **5** (3) (Sept. 1980).
16. G. M. Nijssen, On the gross architecture for the next generation database management systems. In *Information Processing* **77**, edited B. Gilchrist. North-Holland, Amsterdam (1978).
17. G. M. Nijssen, Modelling in data base management systems. In *EURO IFIP 79*, edited P. A. Samet, North-Holland, Amsterdam (1979).
18. T. W. Olle *et al.* (eds). *Information Systems Design Methodologies: A Comparative Review*. North-Holland, Amsterdam (1982).

19. A. B. Saha, Private communication (Dec. 1983).
20. P. M. Stocker, Canonical schemata, canonical queries, and standardisation. In *Infotech State-of-the-art Report: Database*, Series 9, no. 8. Pergamon-Infotech Ltd, Maidenhead, Berkshire, England (1981).
21. P. M. Stocker and R. Cantie, A target logical schema: the ACS. *Proc. 9th VLDB Conference, Florence, Italy* (Oct. 1983).
22. P. M. Stocker, Private communication (May 1984).
23. J. M. Smith and D. C. P. Smith. Database abstractions: aggregation and generalization. *ACM TODS* **2** (2) (June 1977).
24. G. M. A. Verheijen and J. van Bekkum. NIAM: An information analysis method. In Ref. 18.

Conference

BNCOD-5. Fifth British National Conference on Databases, University of Kent at Canterbury in association with The British Computer Society, July 14th–16th 1986.

Call for Papers and Advance Notice

This is the fifth in the series of BNCOD conferences which are primarily designed to provide a forum for British research work in the database field. The 1986 conference is being organised by the University of Kent at Canterbury.

Papers are invited on any aspect of databases and database systems, in particular new developments. Papers related to the following topics will be particularly welcome:

- Transition from Databases to Knowledge Bases
- Integrating Graphics and Text in Databases
- Advanced User Interfaces to Database Systems
- New Database Hardware Architectures
- Distributed Databases (Local Area as well as Wide Area)
- Electronic Office Systems
- Integration of Database Techniques with other areas of Computer Science

Research papers and those based on user experience are welcome.

If you wish to contribute a paper or hope to attend the conference, please write to the conference organiser at the address below for further details:

Dr Elizabeth Oxborrow, BNCOD-5,
Computing Laboratory,
University of Kent at Canterbury,
Canterbury, Kent CT2 7NF.

Intending contributors must submit an abstract (100–200 words) by 1st November 1985 or as soon as possible thereafter.