Fully Comminication Oriented NIAM

G. P. Bakema, J. P. C. Zwart, H. van der Lek

NIAM-ISDM working conference 1994

Abstract

The basic NIAM philosophy is: information analysis intends to model the communication about a certain Universe of Discourse (UoD) but does not intend to model the UoD itself. Consequently all existing elements in NIAM must be consistent with this philosophy. If they are not they must be redefined. Furthermore NIAM must be extended with new elements, such as the complete redundancy free recording of the structure of the declarative sentences spoken by the user expressing (elementary) facts about the UoD. This implies also the need to be able to model complex identification structures occurring in this communication. In this paper we present Fully Communication Oriented NIAM (FCO-NIAM) as a possible solution covering all the above-mentioned desirability's.

We use a generic FCO-NIAM metagrammar (GenMG), which not only enables us to treat both FCO-NIAM and the Relational Model from a single point of view but also to design a simple CASE-tool architecture. This architecture allows the registration of information grammars (IG's) in these Data Models and the transformation of NIAM IG's into relational schemata including the sentence structures via simple updates in the population of the GenMG.

Presently FCO-NIAM is both taught in college-level educations and used in practice on a broad scale in the Dutch NIAM-scene.

1 Introduction

In a previous paper (Van der Lek, Bakema & Zwart, 1992) we presented Communication Oriented NIAM (CO-NIAM) as the most interesting representative of Communication Oriented Information Modeling. An English translation of this paper, titled *Unifying object types and fact types: a practically and didactically productive theory*, was handed out at the NIAM gUIde working conference 1993 held in Utrecht in the Netherlands. It is available from the authors on demand. The present paper deals with Fully Communication Oriented NIAM (FCO-NIAM). Like the previous paper it is written in the heuristic and example based style typical of didactical materials on the college-level. The ideas developed in both papers were given a formal basis in (Van der Lek, 1993).

1.1 Fundamental principles

The following set of structural principles was the starting point for our work (besides the basic set of methodological NIAM-principles such as using natural language, concrete examples, user interviews and prescription based information modeling):

P1:Communication principle:

> ***The purpose of information analysis is not to model the structure of the Universe of Discourse (UoD) itself but to model the structure of the communication about the UoD by the users.***
>
> The product of an information analysis is an Information Grammar (IG), which formally describes on a type level the structure of the relevant fact stating sentences in the user communication about the UoD.

P2': Conceptualization principle:

> ***An IG exclusively models aspects of the user communication the UoD.***

P2": 100 percent principle:

> ***An IG deals with all the aspects of the user communication about the UoD.***

> The formulations of principles P2' and P2" are rephrasings in view of P1 of principles with the same name in (Griethuyzen, 1982). We can summarize P2' and P2" in one principle P2:

P2: 100 percent conceptualization principle:

> ***An IG models all the aspects of the user communication about the UoD and nothing but the user communication about the UoD.***

> See (Nijssen & Halpin, 1989, p. 11).

P3: Redundancy free modeling principle:

> ***An IG models the user communication about the UoD in a redundancy free way.***

> Each aspect of the communication about the UoD may appear only once in an IG.

These principles imply that an information grammar (IG) should model the structure of the fact stating user sentence types also (at least for elementary facts) and in a redundancy free way. The main aspect of our work is an attempt to incorporate these principles in NIAM in a consistent way, because we felt strongly that in traditional NIAM (Nijssen & Halpin, 1989) these principles had not been carried through completely. The way in which we did this in our previous paper was guided by three more principles:

P4: Unification principle:

> ***All non-lexical object types are nominalizations of fact types.***

> This principle solves the most serious violation of the communication principle. It ensures essentially that all non-lexical object types are populatable constructs, unifying fact types and non-lexical object types (including subtypes) into a single concept.

P5: Substitution principle

> ***Elementary user sentences can be regenerated from the IG plus its label population (LP) by substituting either object type expressions (OTE's) or labels in the roles of fact type expressions (FTE's).***

> We imposed this principle on ourselves in order to enable the user to verify the correctness of the modeled declarative sentences (i.e. fact expressions of elementary facts). The separate treatment of OTE's apart from FTE's is a direct consequence of the redundancy free modeling principle (P3).

P6: Generic principle

> ***A Generic Meta Grammar (GenMG) is used which can contain IGs in various data models (NIAM, Relational Model, ...) as its population.***

We adopted this principle for theoretical, practical and didactical reasons; theoretical: NIAM and the Relational Model use different terminologies, yet have a lot in common; practical: improvement of CASE-tool architecture; didactically: teaching information systems methodologies in a generic way. In our previous paper we showed how a relational representation of the GenMG allows us to generate a relational schema from a CO-NIAM IG via ordinary updates on the GenMG population. This is accomplished by the Group, Lexicalize and Reduce (GLR) algorithm.

1.2 Outline of the present paper

The three principles P4, PS and P6 will not be elaborated on again here. We ended our previous paper naming a few subjects for further development. From these the following three points will be worked out in the present paper:

- Inclusion of mandatory role constraints in the GenMG.

- Recording sentence types (more specific: fact type expressions) and object type expressions in the GenMG and generating database sentences (i.e. expressions of stored facts) in natural language.

- A better linguistic anchor for nominalization (i.e. identification structures for object types and the way they can be used in the substitution process).

The incorporation of FTE'S, OTE's and LP's into IG's means that these constructs have to be included in the GenMG as populations of appropriate fact types. Only after accomplishing this we were entitled to call our model Fully Communication Oriented NIAM (FCO-NIAM). So in our FCO-NIAM CASE tool based on principles PI to P6 we have:

- Automatic updates of GenMG populations under the GLR-algorithm.

- Automatic regeneration of elementary user sentences from a given IG plus its LP.

In paragraph 2 we will exemplify information modeling in FCO-NIAM and show how to incorporate FTE'S, OTE's and LP's in IG's. We also demonstrate how to regenerate user sentences from the IG plus its LP.
In paragraph 3 we will treat the Group, Lexicalize and Reduce algorithm (GLR algorithm) and illustrate the behavior of FTE'S, OTF-'s and LP's under this algorithm.

In paragraph 4 we will discuss some extensions to NIAM concerning complex identification structures encountered in practice (generalization, recursive identification structures and set and sequence types). These extensions have been proposed by others as well. We show how these identification structures can be modeled in FCO-NIAM and we indicate their treatment in relational schema derivation.

In paragraph 5 we will consider specialization (subtyping). We illustrate FCO-NIAM subtype modeling and introduce a novel way of subtype identification, using an example which clarifies part of the structure of our GenMG. Unless stated otherwise all features in paragraphs 4 and 5 are presently supported in the prototypical CASEtool built by our students.

In paragraph 6 we comment on the educational and practical impact of FCO-NIAM.

2. From sentences to an information grammar and vice versa

The purpose of this paragraph is to illustrate the consequences of principles P4, PS and P6 for the structure and method of FCO-NIAM, without going into methodological details.

2.1 From sentences to an information grammar

Figure 2. 1 shows a table containing information to be analyzed. The analysis takes place in a dialogue between the analyst and an expert user. The user is requested to verbalize the facts in the table. This is standard NIAM practice (Nijssen & Halpin 1989, Nijssen & Schouten 1993, Nijssen 1993). We call these verbalizations of facts *sentences or fact expressions* (these terms are synonymous). A few example sentences: "Floor 1 exists." (FE1.1), "Floor 2 exists." (FE1.2), "There are 2 emergency exits on floor number 1." (FE2.1), "There are 0 emergency exits on floor number 2" (FE2.2), "There is a room 2. 1." (FE3), "Room 2.1 has 20 seats." (FE4), "Room 2.1 is equipped with a blackboard." (FE5), "We have a blackboard." (FE6.1) and "We have an lcd." (FE6.2). The user either expresses all facts spontaneously or is prompted by the analyst (for instance FE6.2).

FLOOR	# EXITS	ROOM	# SEATS	EQUIPMENT
1	2	1.1	20	blackboard overhead projector
.	.	1.2	30	---
2	0	2.1	20	Blackboard
3	2	3.2a	--	---

Figure 2.1: Floors and rooms

In the dialogue analyst and user group sentences of the same type together. FE2.1 and FE2.2 clearly express two facts of the same *fact type.* FE3, FE4 and FE5 all belong to different fact types. These fact types must be given names. This is standard NIAM practice as well. In the example: FE1.1 and FE1.2 belong to fact type FLOOR, FE2.1 and FE2.2 belong to fact type EMERGENCY EXITS, FE3 belongs to ROOM, FE4 belongs to CAPACITY, FE5 belongs to EQUIPMENT and FE6.1 and FE6.2 belong to FACILITY. In Figure 2.2 these fact types are listed together with one of their sentences.

Analyst and user next classify sentence parts as either *labels* or *object expressions (OE's).* If a sentence part is classified as a label then the name of the *lexical object type (LOT)* the label belongs to must be given. If a sentence part is classified as an OE then it is a nominalized fact and so the name of the corresponding *non-lexical object type (NOLOT)* (or nominalized fact type) must be given (qualification step).

The OE must then be analyzed further: it may contain other OE's or labels. OE's are grouped into types called *object type expressions (OTE's)* the same way as FE's are. The remaining unclassified part of the sentence is called the *predicate.* The predicate together with *place holders* (indicators for the places where labels or OTE's are to be filled in) is called a *fact type expression* (FTE) or *sentence type (ST).* A fact type then is the collection of all FTE's with the same meaning (according to the user) and the same place holder pattern (Van der Lek, Bakema & Zwart, 1992). In fact types the place holders are traditionally called *roles.*

Legenda:
Names of fact types en object types are in CAPITALS.
< >: role. A role number is added for easy reference.
FE: Fact Expression (sentence) FTE: Fact Type Expression
OE: Object Expression OTE: Object Type Expression
Label: single underlining OTE: double underlining

FLOOR:
FE1.2: "Floor 2 exists."
|
FLOOR NUMBER
FTE1: "Floor <FLOOR NUMBER> exists."

EMERGENCY EXITS:
FE2.2: "There are 0 emergency exits on floor number 2.
| |
NUMBER FLOOR
FTE2: "There are <NUMBER> emergency exits on <FLOOR>.

Treating floor number 2 from FE2.2 yields:
OE1: 'floor number 2' (nominalization of FE1.2)
|
FLOOR NUMBER
OTE1.1: 'floor number <FLOOR NUMBER>' (belongs to FLOOR)

From here on we abbreviate the procedure a little in an obvious way.

ROOM:
FE3: "There is a room 2.1."
FTE3: "There is a room <FLOOR>.<ROOM NUMBER>."

Treating 2 from FE3 yields:
OE1.2: '2' (another nominalization of FE1.2)
OTE1.2: '<FLOOR NUMBER>' (belongs to FLOOR)

CAPACITY:
FE4: "Room 2.1 has 20 seats."
FTE4: "<ROOM> has <NUMBER> seats."

Treating Room 2.1 from FE4 yields:
OE3: 'Room 2.1'
OTE3: 'Room <FLOOR>.<ROOM NUMBER>' (belongs to ROOM)
We must now still treat 2 from OE3:
OE1.3: '2' (same as OE1.2 leading to same OTE1.2)

FACILITY:
FE6.1: "We have a blackboard." FE6.2: "We have an lcd.
FTE6: "We have a(n) <FACILITY CODE>."
Note the use of a(n), unifying the words 'a' and 'an' in order to avoid the 'liquistic swamp'.

EQUIPMENT:
FE5: "Room 2.1 is equipped with a(n) blackboard."
FTE5: "<ROOM> is equipped with <FACILITY>."

Treatment of Room 2.1 yields OTE3 again.
OE6: 'a(n) blackboard'
OTE6: 'a(n) <FACILITY CODE>' (belongs to FACILITY)

Figure 2.2: The classification/qualification process

We will not go into the methodology of the classification/qualification procedure here. We only remark that the procedure is different front the classification/qualification procedure in traditional NIAM (Nijssen & Halpin 1989), yet not unlike it in an algorithmic sense. The results of this classification/qualification for each fact type of the example UoD are given in figure 2.2.

A remark convening the terminology used here: In previous papers (Van der Lek, Bakema & Zwart 1992; Van der Lek 1993A) we used the wordt 'formulation' where we use 'expression' here, because we consider it better English and because it stresses the point that sentences putting facts into words are the means by which we communicate (express) those facts. This is also why we prefer the term 'fact type expression' to 'sentence type' here in contrast to our previous paper, although we already noted the synonymy (hyponymy, really, since not all sentences relevant in to UoD express facts) of these terms. Here and there we probably won't be able to avoid mixing them.

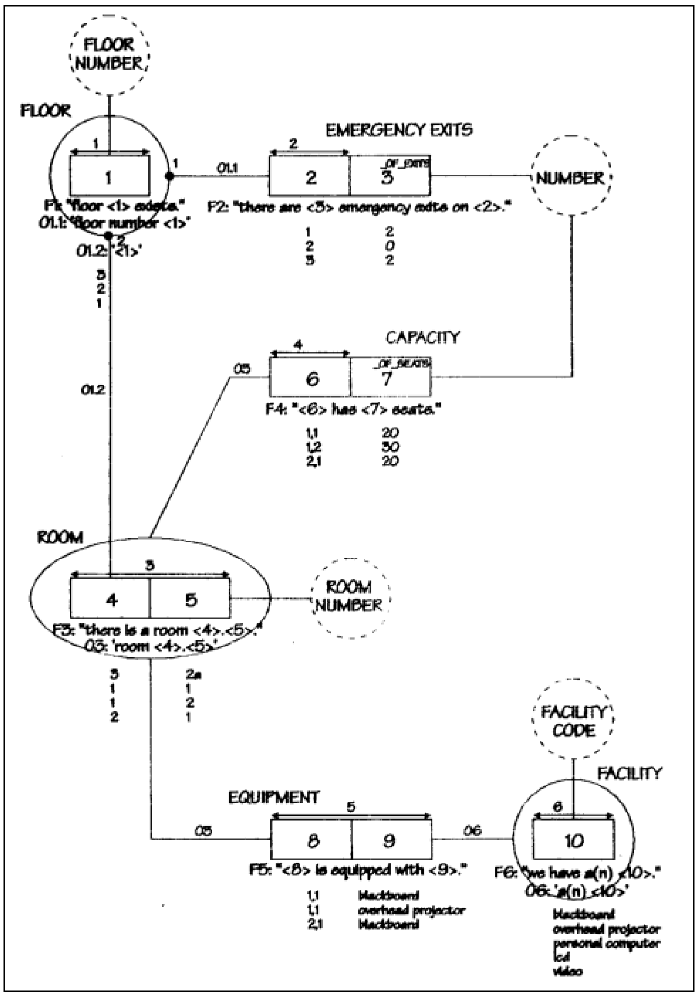


Figure 2.3: FCO-NIAM IG + LP

After the classification/qualification a NIAM diagram can be drawn (Figure 2.3). These diagrams are quite like traditional NIAM diagrams, differing from these in the following aspects:

- All non-lexical object types (NO-LOT'S) are nominalized fact types.
- Each fact type has at least one fact type expression (FTE) written underneath the roles. In the diagram, the abbreviation F is used instead of FTE.
- Each NOLOT has at least one object type expression (OTE) written underneath its FTE. In the diagram, the abbreviation O is used instead of OTE.

- Next to each role, played by a NOLOT, the identifier of the OTE to be used by that role is written. For example: role 2 uses O1. 1, but role 4 uses O1.2. O1.1 and O1. 2 are different OTE's belonging to FLOOR.
- So-called prefixes, postfixes or insteadfixes may be freely added to roles. In figure 2.3 a postfix _OF_EXITS has been added to role 3 and a postfix _OF_SEATS to role 7. Their only purpose is to enable automatic generation of meaningful columnnames after transformation of the information grammar to a relational schema. In the example columnnames NUMBER_OF_EXITS and NUMBER_OF_SEATS will be generated. Prefixes are added to the front of a standard column name and postfixes to the end. Insteadfixes replace a standard column name.

To the diagram the standard constraints, such as uniqueness constraints, mandatory role constraints etc. must be added. These do not concern us here, as the standard NIAM procedure can he employed. Figure 2.3 has them drawn in. As for subtype constraints: see § 5. A few remarks must be made, however:

- The meaning of mandatory role constraints is in FCO-NIAM slightly different from traditional NIAM. Example: mandatory role constraint 2 (on role 4) means, that the set of role values of role 4 must he equal to the set of role values of role 1. This solves inorthogonalities and an inconsistency with respect to mandatory role constraints encountered in traditional NIAM.
- More than one OTE-identifier may be written next to a role. They may all be used by that role, although in general there are other restrictions depending on the situation (see § 4).
- A more sophisticated variant of the procedure to write OTE-identifiers next to roles is called *subtle substitution:* it enables the analyst to specify in which FTE a certain OTE is to be substituted. For example: if fact type EMERGENCY EXITS would have two FTE's called F2.1 and F2.2, we might specify the use of O1.1 in F2.1 and O1.2 in F2.1 by writing next to role 2 the codes O1.1:F2.1 and O1.2:F2.1. This is necessary in order to model user communication correctly and is implemented in our CASE-tool.

2.2 From an information grammar to sentences

It is very easy to (re)generate the original user sentences from the Information grammar (IG) and its label population (LP). We use the algorithm of the substitution principle in order to do that:

a) Start with a fact type expression (FTE), and one of the tupels of the population of the corresponding fact type.

b) Its place holders must be filled in.

b1) If the corresponding role is played by a lexical object type (LOT), then fill the label in directly.

b2) If the corresponding role is played by a non-lexical object type (NOLOT), then fill in one of the object type expressions (OTE's) that may he used by that role. As each OTE will contain roles itself, continue with step b for the OTE.

This recursive process will end when all labels are finally filled in. As an example we will generate one sentence from fact type CAPACITY, which is part of figure 2.3.

a) The fact type expression is F4: " < 6 > has < 7 > seats." The tupel has '2, 1' as role value of role 6 (and '20' as role value of role 7).

b1) < 7 > is played by a LOT. '20' may he directly substituted.
Result: " < 6 > has 20 seats."
b2) < 6 > is played by a NOLOT. We use O3: 'room < 4 >. < 5 >'.
Result: "room < 4 > . < 5 > has 20 seats."

b1) < 5 > is played by a LOT. We substitute label '1' directly.
Result: "room < 4 >.1 has 20 seats."
b2) < 4 > is played by a NOLOT. We use O1.2: '< 1 >'.
Result: "room < 1 >.1 has 20 seats."
b1) < 1 > is played by a LOT. We substitute label '2' directly.
Result: "room 2.1 has 20 seats."

When we capitalize the first letter we have indeed a user sentence:

FE4: "Room 2.1 has 20 seats".

This process can and has been completely automated. Our CASE-tool generates these sentences from the IG and LP instantly.

A standard way of showing all sentences modeled in an IG plus LP is depicted in figure 2.4. Here all OTE's have been substituted until only roles played by LOT's are left. Instead of the role numbers the names of the LOT's are shown (as in figure 2.2). To each of these sentence types the label population is added. This form provides a convenient shorthand for the <u>user</u>, who will not be familiar with the NIAM diagram.

F1: "floor <FLOOR NUMBER> exists."

1
2
3

F2:"there are <NUMBER_OF_EXITS> emergency exists on floor number <FLOOR NUMBER>."

30	1
0	2
20	3

F3:"there is a room <FLOOR NUMBER>.<ROOM NUMBER>."

1	1
1	2
2	1
3	2a

F4:"room <FLOOR NUMBER>.<ROOM NUMBER> has <NUMBER_OF_SEATS> seats."

1	1	20
1	2	30
2	1	20

F5:"room <FLOOR NUMBER>.<ROOM NUMBER> is equipped with a(n) <FACILITY CODE>."

1	1	blackboard
1	1	overhead projector
2	1	blackboard

F6:"we have a(n) <FACILITY CODE>."

blackboard
overhead projector
personal computer
lcd
video

Figure 2.4: Sentence types + population

3. The GLR-algorithm for relational schema generation

The purpose of this paragraph is to illustrate how an elementary FCO-NIAM information grammar (including FTE'S, OTE's and LP's) can be transformed into a relational schema by a series of updates on the population of the generic metagrammar.

3.1 Tupel numbers and tupel references

In the generic metagrammar (GenMG) we use *tupel numbers.* Each tupel can be identified by giving the name of the fact it belongs to together with its tupel number (unique per fact type). This facilitates the recording and handling of the tupels. In an FCO-NIAM IG we write tupel numbers followed by a colon in front of the tupels (figure 3.1). In addition to this we use so-called *tupel references.* The role values of every role played by a non-lexical object type (NOLOT) are not (combinations of) labels, but are the tupel numbers of the tupels in the population of the NOLOT playing the role. For example: in figure 3.1 in fact type ROOM the role value of role 4 in tupel 2 is [3]. This refers to tupel number 3 from fact type FLOOR, which has as role value '1', a label identifying the first floor. Compare this to the corresponding tupels in figure 2.3). In an FCO-NIAM IG we enclose the tupel references in square brackets (Figure 3.1). The role values of all roles played by lexical object types (LOT's) are simply the labels themselves. Please note this does not affect in any way the substitution algorithm (see § 2.2). It is just another way of writing down a population of an FCO-NIAM IG: more compact and redundancy-free.

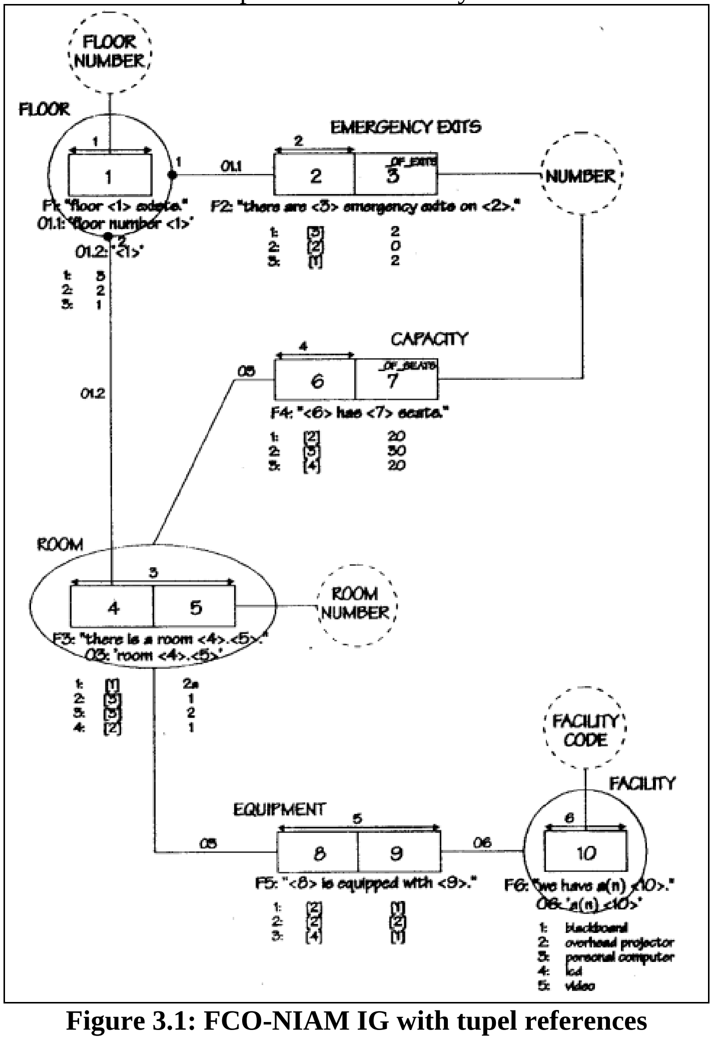


Figure 3.1: FCO-NIAM IG with tupel references

3.2 The generic metagrammar (GenMG) in relational form

Here we present the main part of our generic metagrammar (GenMG) in relational form. We have the complete GenMG in FCO-NIAM form of course, but we want to concentrate on the main parts of it in its most concise and conveniently arranged manifestation. Please note that because the GenMG is itself also an FCO-NIAM information grammar (IG) it can be populated with itself. This autopopulation can then be subjected to the GLR algorithm in order to transform it into its relational form. This again stresses the point that the GenMG can contain as population IG's in various data models, FCO-NIAM and the Relational Model in particular. If an MG is to be generic for a set of Data Models, it must contain the union of all the model-specific fact/object types but only the intersection of all the model-specific constraints. In order to ensure that a certain population will represent a correct IG in a certain Data Model, we temporarily 'turn on' the model-specific constraints which do not belong to the intersection. After this validation we 'turn' those specific constraints 'off' again. The GLR-algorithm actually works in this way, turning off the FCO-NIAM specific constraints and turning on those specific for the Relational Model.

Table OBJECTTYPE/FACTTYPE

OT/FT_NAME	CLASSCODE	DATATYPE_NAME	TOT_LENGTH	DEC_LENTH
FLOOR NUMBER	L	SMALLINT	.	.
FLOOR	N	.	.	.
EMERGENCY EXIT	F	.	.	.
NUMBER	L	INTEGER	.	.
CAPACITY	F	.	.	.
ROOM	N	.	.	.
ROOM NUMBER	L	CHAR	6	.
FACILITY CODE	L	VARCHAR	40	.
FACILITY	N	.	.	.
EQUIPMENT	F	.	.	.

Table OTE IN ROLE

OTE_CODE	OTE_NO	ROLE
O	1.1	2
O	1.2	4
O	3	6
O	3	8
O	6	9

Table ROLE

ROLE_NO	PART_OF_OT/FT	COLUMN_FIX	PLAYED_BY_OT/FT	OP_INDIC
1	FLOOR	.	FLOOR NUMBER	NO
2	EMERGENCY EXITS	.	FLOOR	NO
3	EMERGENCY EXITS	_OF_EXISTS	NUMBER	NO
4	ROOM	.	FLOOR	NO
5	ROOM	.	ROOM NUMBER	NO
6	CAPACITY	.	ROOM	NO
7	CAPACITY	_OF_SEATS	NUMBER	NO
8	EQUIPMENT	.	ROOM	NO
9	EQUIPMENT	.	FACILITY	NO
10	FACILITY	.	FACILITY CODE	NO

Table EXPRESSION

EXP_CODE	EXP_NO	OT/FT
F	1	FLOOR
F	2	EMERGENCY EXITS
F	3	ROOM
F	4	CAPACITY
F	5	EQUIPMENT
F	6	FACILITY
O	1.1	FLOOR
O	1.2	FLOOR
O	3	ROOM
O	6	FACILITY

Table EXPRESSION PART (only partly shown)

EXP_CODE	EXP_NO	PART_NO	ROLE	TEXT
F	1	1	.	floor
F	1	2	1	.
F	1	3	.	exists

Table PRIMARY KEY

UC	PRIM_INDIC
1	YES
2	YES
3	YES

Table UNIQUENESS CONSTRAINT

UC_NO	ROLE
1	1
2	2
3	4

F	4	1	6	.
F	4	2	.	has
F	4	3	7	.
F	4	4	.	seats
O	1.1	1	.	floor number
O	1.1	2	1	.
O	1.2	1	1	.
O	3	1	.	room
O	3	2	4	.
O	3	3	.	.
O	3	4	5	.
...	...	...	...	...

4	YES
5	YES
6	YES

4	6
5	8
6	9
6	10

Table POPULATION (only party shown)

ROLE	POS_TUPEL_NO	REF_TUPEL_NO	LABEL
1	1	.	3
1	2	.	2
1	3	.	1
4	1	1	.
4	2	3	.
4	3	3	.
4	4	2	.
5	1	.	2a
5	2	.	1
5	3	.	2
5	4	.	1
6	1	2	.
6	2	3	.
6	3	4	.
...	...	...	......

Table MANDATORY ROLE CONSTRAINT

MRC_NO	ROLE
1	2
2	4

Table SUBSET CONSTRAINT

SC_NO	FROM_ROLE	TO_ROLE
.	.	.

Figure 3.2: GenMG populated with elementary IG

In figure 3.2 we show the main part of the GenMG in relational form, populated with the FCO-NIAM IG of figure 3.1. We will now comment briefly on all the tables. Columns with an obvious meaning will not be discussed.

Table OBJECTTYPE/FACTTYPE.

In FCO-NIAM the three concepts LOT, fact type and NOLOT are unified in the single Concept *objecttype/facttype*. In the FCO-NIAM form Of the GenMG the three 'old' concepts are subtypes of OBJECTTYPE/FACTTYPE (see § 5).

- CLASSCODE: list to which class a particular objecttype/facttype belongs. 'L' stands for LOT, 'N' stands for Nominalized fact type, .'F' stands for a Fact type that does not play roles (is not nominalized). These three codes suffice to define the entire subtype structure.

- DATATYPE_NAME, TOT_LENGTH and DEC_LENGTH: optional columns not necessary for FCO-NIAM or the Relational Model, but convenient when generating DDL-statements for a relational schema. TOT_LENGTH specifies the maximum length of character strings or decimal figures. DEC_LENGTH specifies the number of figures after the decimal point.

Table ROLE.

- ROLE_NO: a role number is a unique role-identifier. (Other identification schemes are possible, such as unique number per fact type, but we considered this the most convenient choice.)

- PART_OF_OT/FT specifies which objecttype/facttype the role is a part of. For instance: in figure 3.1 role 4 is a part of objecttype/facttype ROOM.

- PLAYED_BY_OT/FT specifies which objecttype/facttype plays the role. For instance: in figure 3.1 role 4 is played by objecttype/facttype FLOOR.

- COLUMN_FIX: an optional column allowing pre- post- or insteadfixes which will be added to or override the standard column names generated by our CASE-tool.

- OP_INDIC: specifies whether the role is optional or not. In an elementary traditional NIAM IG, all roles are NOT optional. In an FCO-NIAM IG, even in an elementary one, roles may be optional under certain conditions (see § 4). The GLR-algorithm will mark appropriate roles as optional ('YES' in OP_INDIC) during its execution, because in the Relational Model roles (called 'columns' in that Model) must sometimes be optional.

Table UNIQUENESS CONSTRAINT.

- UC_NO: uniqueness constraints are identified by a unique number.

A verbalization of two tupels about the same constraint: "Uniqueness constraint 3 concerns (among others) role 4." "Uniqueness constraint 3 concerns (among others) role 5."

Table PRIMARY KEY.

A fact type can have more than one uniqueness constraint. In such cases, the GLR-algorithm requires to assign one as the primary key (if that is not possible, the FCO-NIAM IG cannot be transformed to a relational schema which meets the entity integrity requirement. This can happen because FCO-NIAM is more powerful than the Relational Model. Standard transformations are being worked out to arrive at a relational schema by changing the communication (see § 4). In the example of figure 3.1 this does not occur, so all uniqueness constraints are marked as primary.

Table MANDATORY ROLE CONSTRAINT.

This table is analogous to UNIQUENESS CONSTRAINT.

Table SUBSET CONSTRAINT.

- SC_NO: each subset constraint is identified by a unique number.

- FROM_ROLE and TO_ROLE: FROM_ROLE lists a role whose role values are a subset of the role in the same tupel listed under TO_ROLE. One subset constraint can have two or more of such role pairs (see figure 3.6).

In this paper equality constraints are treated as pairs of subset constraints in opposite directions. (Exclusion constraints lead to a similar table not discussed in this paper.)

Table EXPRESSION.

- EXP_CODE: lists the kind of expression: F for Fact type expressions, O for Object type expressions.
- EXP_NO: together with the expression code the expression number identifies an expression.
- OT/FT: lists the objecttype/facttype the expression belongs to.

Table OTE IN ROLE.

- OTE_CODE and OTE_NO are the same as EXP_CODE and EXP_NO in table EXPRESSION. The column names have been changed to stress that only OTE's may be listed here.
- ROLE lists the role that uses the OTE in the same tupel.

An example verbalization: "Object type expression O1.1 may be used by role 2. "

Table EXPRESSION PART.

EXP_CODE, EXP_NO and PART_NO together identify part of an expression. An expression part can either be a ROLE or a piece of TEXT. An expression is made up of a sequence of such parts. For instance: the three tupels with EXP_CODE 'F' and EXP_NO '1' correspond to F1: "floor <1> exists".

Table POPULATION.

- POS_TUPEL_NO lists actual tupel numbers (see § 3.1).
- REF_TUPEL_NO lists tupel references (see § 3.1).
- LABEL lists actual labels.

Example verbalizations:

"The role value of role 1 in tupel number 1 is the label '3'."
"The role value of role 4 in tupel number 4 is the tupel reference [2]."

3.3 The example IG as population of the GenMG under the GLR-algorithm

The example IG as population of the GenMG

Figure 3.2 above shows the GenMG populated with the IG from figure 3.1. Datatypes have been added. Please note that it is possible to regenerate the entire figure 3.1 (population included) from the tupels in these tables, excluding only the layout of the diagram.

The example IG as population of the GenMG after grouping

Basically, the grouping part of the GLR-algorithm does the following:

1. Mark roles with the following properties: they have a uniqueness constraint concerning only that one role; they are played by a NOLOT; they are derivable in case of subtypes (§ 5).
2. Cancel each role thus marked, absorbing all other roles from the fact type the role was in into the objecttype/facttype that played the role. Cancel the original fact type the role belonged to.
3. Update remaining constraints, FTE's OTE's and the population.

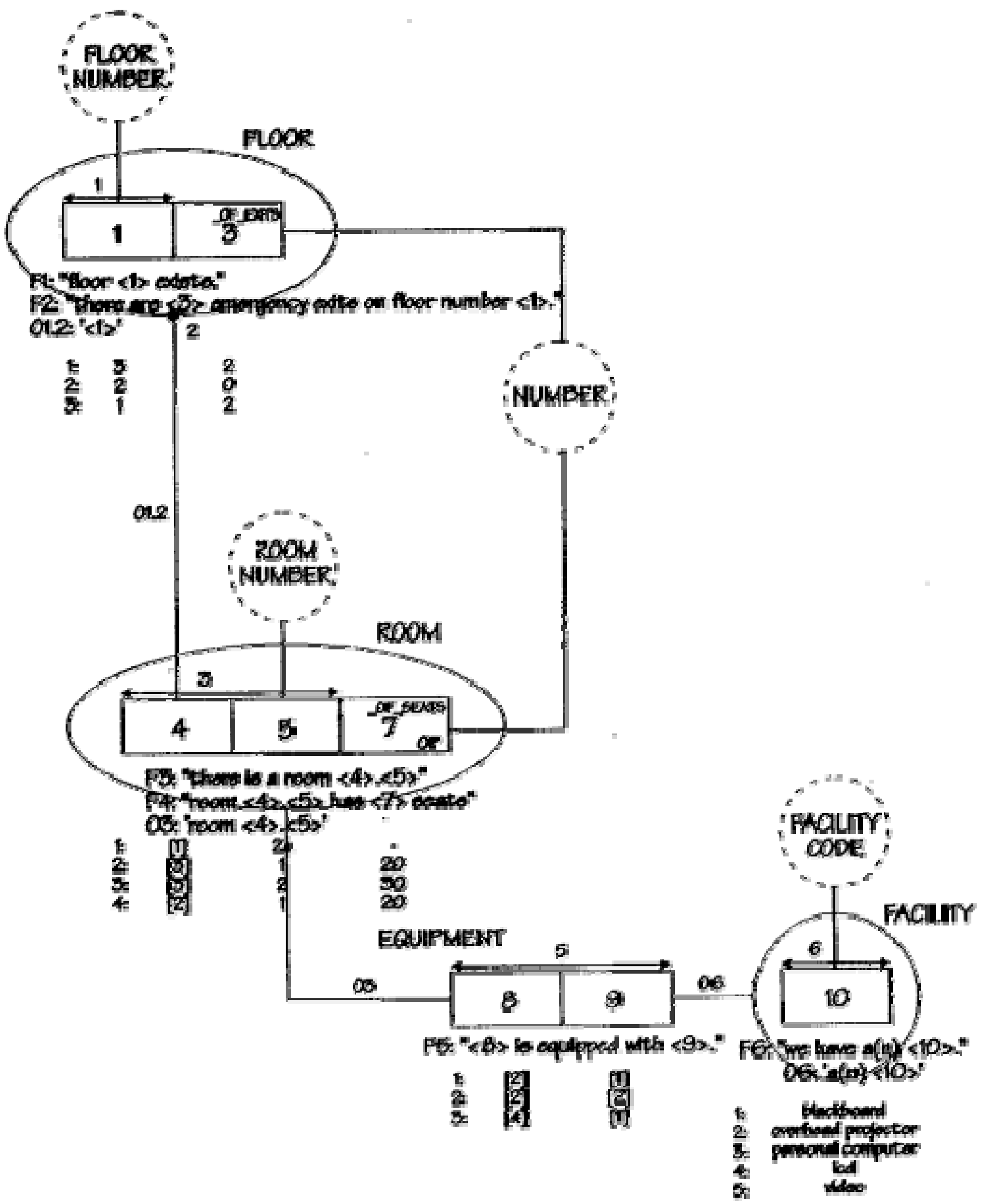


Figure 3.3: IG after grouping

The result of the grouping operation in diagram-form is shown in figure 3.3, and the corresponding population of the GenMG in figure 3.4. Please note the transformation is done only on the population of the GenMG; figure 3.3 is drawn as an illustration afterwards.

We draw your attention to the following points:

- Roles 2 and 6 have disappeared, along with the fact types which contained them (EMERGENCY EXITS and CAPACITY). Role 3 is absorbed in fact type FLOOR, remaining non-optional because of mandatory role constraint 1, which disappears after the grouping operation. Role 7 is absorbed in fact type ROOM, becoming optional since there was no mandatory role constraint exclusively belonging to role 6.

- FTE's F2 and F4 are moved to FLOOR and ROOM respectively, with the appropriate OTE's substituted into the place holders. Of course no FTE's may ever disappear because that would result in a loss of information. We keep all FTE's elementary, even when a fact type itself is no longer elementary after the grouping operation. The reason is we want to avoid language-specific syntactical rules (linguistic swamp) for conjunctions of FTE's. Such a non-elementary fact type will then have more than one FTE. Any OTE that is no longer necessary (for instance O1.1) is canceled. The population of EMERGENCY EXITS and CAPACITY is integrated with the populations of FLOOR and ROOM.
- Note the occurrence after grouping of a null-value (designated by a hyphen) in tupel 1 of ROOM. Null-values are simply not recorded in the GenMG, only shown as hyphens in the diagram for ergonomic reasons. See table POPULATION, the three tupels concerning role 7: there is no POS_TUPEL_NO equal to '1'.

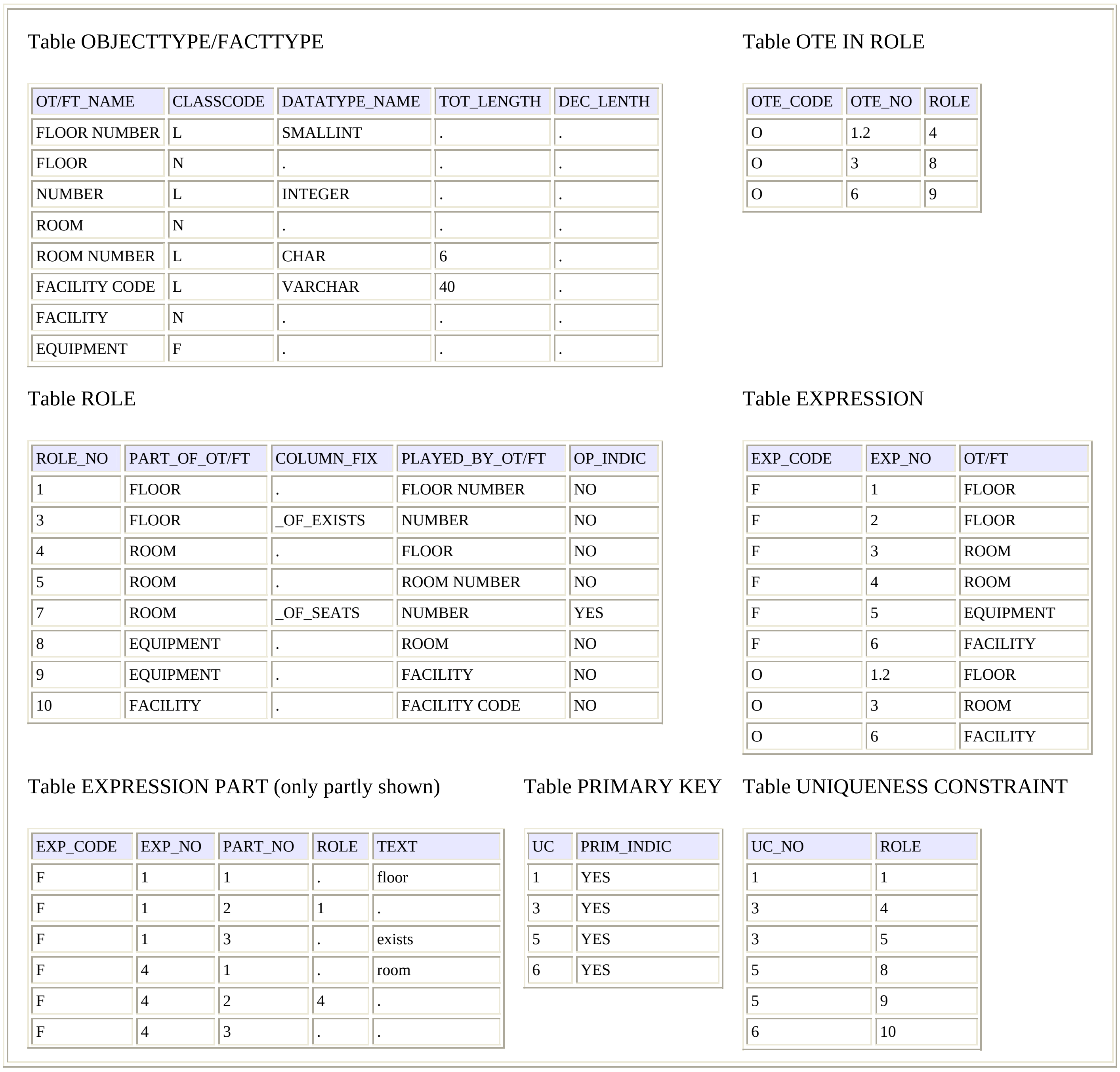


Table OBJECTTYPE/FACTTYPE

OT/FT_NAME	CLASSCODE	DATATYPE_NAME	TOT_LENGTH	DEC_LENTH
FLOOR NUMBER	L	SMALLINT	.	.
FLOOR	N	.	.	.
NUMBER	L	INTEGER	.	.
ROOM	N	.	.	.
ROOM NUMBER	L	CHAR	6	.
FACILITY CODE	L	VARCHAR	40	.
FACILITY	N	.	.	.
EQUIPMENT	F	.	.	.

Table OTE IN ROLE

OTE_CODE	OTE_NO	ROLE
O	1.2	4
O	3	8
O	6	9

Table ROLE

ROLE_NO	PART_OF_OT/FT	COLUMN_FIX	PLAYED_BY_OT/FT	OP_INDIC
1	FLOOR	.	FLOOR NUMBER	NO
3	FLOOR	_OF_EXISTS	NUMBER	NO
4	ROOM	.	FLOOR	NO
5	ROOM	.	ROOM NUMBER	NO
7	ROOM	_OF_SEATS	NUMBER	YES
8	EQUIPMENT	.	ROOM	NO
9	EQUIPMENT	.	FACILITY	NO
10	FACILITY	.	FACILITY CODE	NO

Table EXPRESSION

EXP_CODE	EXP_NO	OT/FT
F	1	FLOOR
F	2	FLOOR
F	3	ROOM
F	4	ROOM
F	5	EQUIPMENT
F	6	FACILITY
O	1.2	FLOOR
O	3	ROOM
O	6	FACILITY

Table EXPRESSION PART (only partly shown)

EXP_CODE	EXP_NO	PART_NO	ROLE	TEXT
F	1	1	.	floor
F	1	2	1	.
F	1	3	.	exists
F	4	1	.	room
F	4	2	4	.
F	4	3	.	.

Table PRIMARY KEY

UC	PRIM_INDIC
1	YES
3	YES
5	YES
6	YES

Table UNIQUENESS CONSTRAINT

UC_NO	ROLE
1	1
3	4
3	5
5	8
5	9
6	10

F	4	4	5	.
F	4	5	.	has
F	4	6	7	.
F	4	7	.	seats
O	5	1	8	.
O	5	2	.	is equipped with
O	5	3	9	.
O	1.2	1	1	.
O	3	1	.	room
...	...	...	...	...

Table POPULATION (only party shown)

ROLE	POS_TUPEL_NO	REF_TUPEL_NO	LABEL
1	1	.	3
1	2	.	2
1	3	.	1
4	1	1	.
4	2	3	.
4	3	3	.
4	4	2	.
5	1	.	2a
5	2	.	1
5	3	.	2
5	4	.	1
7	2	.	20
7	3	.	30
7	4	.	20
...	...	...	......

Table MANDATORY ROLE CONSTRAINT

MRC_NO	ROLE
2	4

Table SUBSET CONSTRAINT

SC_NO	FROM_ROLE	TO_ROLE
.	.	.

Figure 3.4: GenMG populated with IG after grouping

The example IG as population of the GenMG after lexicalization

In the lexicalization step of the GLR-algorithm all roles played by NOLOTS are treated. After lexicalization all roles are played by LOT's. This is accomplished by 'diverting' the line connecting the role to the NOLOT to the LOT identifying the NOLOT. For instance: the line from role 9 to NOLOT FACILITY will be diverted to LOT FACILITY CODE. sometimes two or more 'diversions' must be made before reaching a LOT. This process also involves the splitting of each role played by a NOLOT into as many roles as are contained inside the NOLOT. Whenever a 'diversion' is made, it generates one subset constraint in case no mandatory role constraint is present or two subset constraints (equivalent to one equality constraint) if a mandatory role constraint is involved. For instance: the diversion of role 9 creates one subset constraint (with SC_NO 4 in table SUBSET CONSTRAINT in figure 3.6). The consequences must he processed in FTE's, population etc.

Figure 3.5 shows the result of this step in diagram form. Figure 3.6 shows the population of the GenMG. We draw your attention to the following points:

- Role 8 has been split into roles 8.1 and 8.2, because there were two roles inside the fact type ROOM. Role 8.1 corresponds to role 4 and 8.2 to 5. The subsequent diversion generates subset constraint 3.
- Diverting role 4 from FLOOR to FLOOR NUMBER generates subset constraints 1 and 2 (forming one equality constraint) because of mandatory role constraint 2.
- No OTE's are left after lexicalization: all were substituted into appropriate FTE's.
- No tupel references are left because all roles are played by LOT's.
- No mandatory role constraints are left because only roles played by NOLOT's can have them.

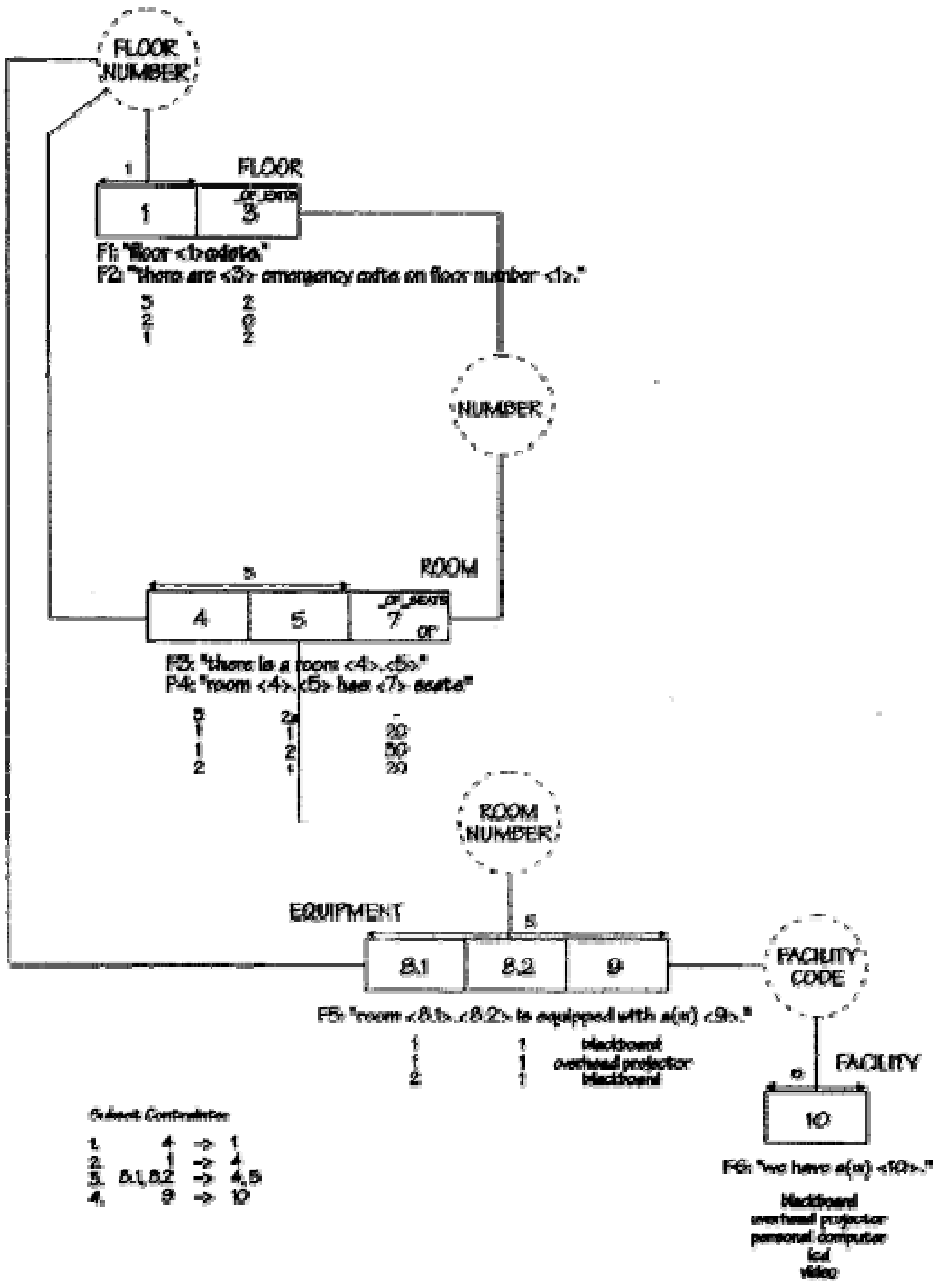


Figure 3.5: IG after grouping and lexicalizing

Table OBJECTTYPE/FACTTYPE

OT/FT_NAME	CLASSCODE	DATATYPE_NAME	TOT_LENGTH	DEC_LENTH
FLOOR NUMBER	L	SMALLINT	.	.
FLOOR	F	.	.	.
NUMBER	L	INTEGER	.	.
ROOM	F	.	.	.
ROOM NUMBER	L	CHAR	6	.
FACILITY CODE	L	VARCHAR	40	.
FACILITY	F	.	.	.
EQUIPMENT	F	.	.	.

Table OTE IN ROLE

OTE_CODE	OTE_NO	ROLE
.	.	.

Table ROLE

ROLE_NO	PART_OF_OT/FT	COLUMN_FIX	PLAYED_BY_OT/FT	OP_INDIC
1	FLOOR	.	FLOOR NUMBER	NO
3	FLOOR	_OF_EXISTS	NUMBER	NO
4	ROOM	.	FLOOR NUMBER	NO
5	ROOM	.	ROOM NUMBER	NO
7	ROOM	_OF_SEATS	NUMBER	YES
8.1	EQUIPMENT	.	FLOOR NUMBER	NO
8.2	EQUIPMENT	.	ROOM NUMBER	NO
9	EQUIPMENT	.	FACILITY CODE	NO
10	FACILITY	.	FACILITY CODE	NO

Table EXPRESSION

EXP_CODE	EXP_NO	OT/FT
F	1	FLOOR
F	2	FLOOR
F	3	ROOM
F	4	ROOM
F	5	EQUIPMENT
F	6	FACILITY

Table EXPRESSION PART (only partly shown)

EXP_CODE	EXP_NO	PART_NO	ROLE	TEXT
F	1	1	.	floor
F	1	2	1	.
F	1	3	.	exists
F	4	1	.	room
F	4	2	4	.
F	4	3	.	.
F	4	4	5	.
F	4	5	.	has
F	4	6	7	.
F	4	7	.	seats
...	...	...	...	...

Table PRIMARY KEY

UC	PRIM_INDIC
1	YES
3	YES
5	YES
6	YES

Table UNIQUENESS CONSTRAINT

UC_NO	ROLE
1	1
3	4
3	5
5	8.1
5	8.2
5	9
6	10

Table POPULATION (only party shown)

ROLE	POS_TUPEL_NO	REF_TUPEL_NO	LABEL
1	1	.	3
1	2	.	2
1	3	.	1

Table MANDATORY ROLE CONSTRAINT

MRC_NO	ROLE
.	.

4	1	.	3
4	2	.	1
4	3	.	1
4	4	.	2
5	1	.	2a
5	2	.	1
5	3	.	2
5	4	.	1
7	2	.	20
7	3	.	30
7	4	.	20
...	...	...	......

Table SUBSET CONSTRAINT

SC_NO	FROM_ROLE	TO_ROLE
1	4	1
2	1	4
3	8.1	4
3	8.2	5
5	9	10

Figure 3.6: GenMG populated with IG after grouping and lexicalizing

The example IG as population of the GenMG after reducing

In this example no reducing can be carried out. In general, reducing means removing redundant fact types. After grouping fact types may exist having both a primary key concerning all roles in the fact type and an equality constraint concerning all roles in the fact type (and as many roles in another fact type). In such a case the fact type can be removed (after moving its FTE to the other fact type). If for instance role 9 would have had a mandatory role constraint in figures 3.1 and 3.3, then in figure 3.5 an equality constraint between roles 9 and 10 would have been generated and fact type FACILITY would be removed, transporting fact type expression F6 to EQUIPMENT.

Alternative diagram for the example IG after the GLR-algorithm

Figure 3.5 actually shows a relational schema with domains(LOT's), tables (fact types), primary keys (equal to certain uniqueness constraints), table columns (roles), NOT NULL indicators (roles without an OP-indicator), references, including foreign keys (a kind of subset constraints tupels (tupels). In addition it shows the user sentence types. A perhaps more familiar way of drawing tables is shown in figure 3.7, which is completely equivalent to figure 3.5. We emphasize that FCO-NIAM, the Relational Model and other Data Models can best be seen as special cases of the Generic Data Model we employ here.

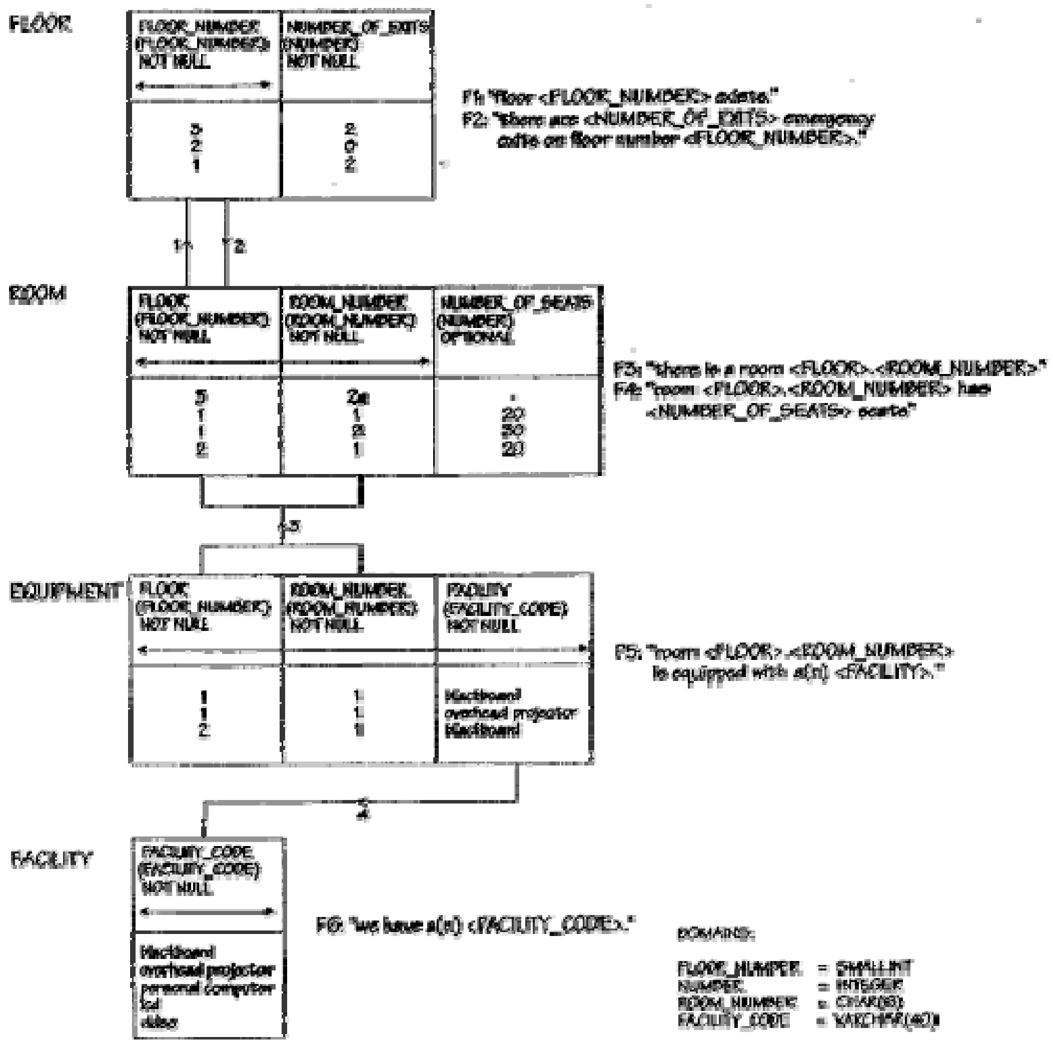

FLOOR
FLOOR_NUMBER (FLOOR_NUMBER) NOT NULL
NUMBER_OF_EXITS (NUMBER) NOT NULL
3 2 1
2 0 2
F1: "floor <FLOOR_NUMBER> exists."
F2: "there are <NUMBER_OF_EXITS> emergency exits on floor number <FLOOR_NUMBER>."
1
2
ROOM
FLOOR (FLOOR_NUMBER) NOT NULL
ROOM_NUMBER (ROOM_NUMBER) NOT NULL
NUMBER_OF_SEATS (NUMBER) OPTIONAL
3 1 1 2
2a 1 2 1
20 30 20
F3: "there is a room <FLOOR>.<ROOM_NUMBER>"
F4: "room <FLOOR>.<ROOM_NUMBER> has <NUMBER_OF_SEATS> seats"
5
EQUIPMENT
FLOOR (FLOOR_NUMBER) NOT NULL
ROOM_NUMBER (ROOM_NUMBER) NOT NULL
FACILITY (FACILITY_CODE) NOT NULL
1 1 2
1 1 1
blackboard
overhead projector
blackboard
F5: "room <FLOOR>.<ROOM_NUMBER> is equipped with a(n) <FACILITY>."
4
FACILITY
FACILITY_CODE (FACILITY_CODE) NOT NULL
blackboard
overhead projector
personal computer
tcd
video
F6: "we have a(n) <FACILITY_CODE>."
DOMAINS:
FLOOR_NUMBER = SMALLINT
NUMBER = INTEGER
ROOM_NUMBER = CHAR(3)
FACILITY_CODE = VARCHAR(40)

Figure 3.7: Relational schema

A few final remarks:

- During execution of the grouping algorithm roles marked for grouping may become optional. In that case under certain conditions their marking will be undone. A complete treatment of this matter would take us too far afield here. There is a connection however with the question whether or not it should be allowed to have multiple identification structures in relational schemata (see also the final remark of § 4.1).
- Subset constraints are usually defined between two sets of roles of equal cardinality: (4,5) -- > -- (8,9), meaning: the set of combinations of role values of roles 4 and 5 must be a subset of the set of combinations of role values of roles 8 and 9. Subset constraints can he generalized to include unions as well: (2U6U9) -- > -- (3U4): the set of the union of the role values of roles 2, 6 and 9 must be a subset of the set of the union of the role values of roles 3 and 4. Simple cases of such subset constraints arise often during execution of the GLR-algorithm (when there is a mandatory role constraint concerning several roles). These simple cases can he incorporated in the table SUBSET CONSTRAINT and dealt with in the reducing step of the GLR-algorithm. The treatment of the most general case is beyond the scope of this paper (it would require a different table structure for the recording of subset constraints).
- The standard way column names are generated is: use the name of the object type which originally plays the role (i.e. before grouping). For the treatment of prepost- and insteadfixes please refer to § 2.1 and § 3.2.

4. The need for complex identification structures

Classic NIAM textbooks like (Nijssen & Halpin, 1989) do not treat the modeling of multiple identification (generalization), variable length identification by ordered or unordered enumeration (sequence and set based identification) or recursive identification. We introduce the general term complex identification for these identification schemes. The 100 percent conceptualization principle (P2 in § 1.1) requires us to be able to model complex identification without introducing extra labels (identifying codes or numbers), even though this is a common device in practice. The substitution principle requires us to be able to regenerate user fact expressions containing complex identification by substitution.

In § 4.1 we discuss a typical example of generalization. In § 4.2 we illustrate the need for recursive identification with an example from a real life case. In § 4.3 we treat set and sequence based identification.

4.1 Generalization

The necessity of incorporating generalization alongside specialization is hardly disputed nowadays. Different treatments exist (for example: Ter Hofstede, 1993, pp. 37-39). The FCO-NIAM way to deal with generalization is the subject of this paragraph. It turns out we don't need to introduce new meta-concepts: our generic metagrammar (GenMG) presented in § 3 can manage this. The classification / qualification process, described in § 2 (figure 2.2), automatically leads to multiple identification structures if they occur in the user sentences. We will not elaborate on the details here, but present a simple example to illustrate the way generalization is modeled in FCO-NIAM instead.

A room rental company identifies rooms by room numbers (room 1, room 2, etc.) or by room names (auditorium, green room, blue room, etc.). Some rooms have both a room number and a room name (room 1 = the green room, room 2 = the blue room), other rooms have only a room number (room 17, etc.) and the auditorium only has a name: auditorium. We overhear parts of dialogues (i.e. communication) between employees and different visitors:

"Where is the meeting of the NIAM Group?"
la: "The meeting of the NIAM Group takes place in room 1."

"Where does the meeting of the NIAM Group take place?"
1b: "The meeting of the NIAM Group takes place in the green room."

"In which room is the meeting of the NIAM Group?'
lc: "The meeting of the NIAM Group takes place in room 1 also called the green room."

"Where are the meetings of the Dutch Railway Company and the Trade Union?"
2: "The meeting of the Dutch Railway Company takes place in room 17."
3: "The meeting of the Trade Union takes place in the auditorium."

The fact expressions are identified by a number, possibly followed by a letter a, b or c if there are different expressions of the same fact. We have underlined the object expressions already. The classification / qualification process (compare with figure 2.2) leads to the following fact types, fact type expressions, object type expressions, label type and object

ORGANIZATION:

FTE1: "there is an organization named <ORGANIZATION NAME>."
OTE1: 'the <ORGANIZATION NAME>'

ROOM:

> FTE2a: "the room <ROOM NUMBER> exists.'
> OTE2a: 'room <ROOM NUMBER>'
> FTE2b: "the room <ROOM NAME> exists."
> OTE2b: 'the <ROOM NAME>'
> FTE2c: "room < ROOM NUMBER > is the same room as the < ROOM NAME >."
> OTE2c: 'room < ROOM NUMBER > also called the < ROOM NAME >'

MEETING:

> FTE3: "the meeting of < ORGANIZATION > takes place in < ROOM >."

From this the FCO-NIAM information grammar presented in figure 4. 1 follows. The sentences 1a, 1b, 1c, 2 and 3 can easily be regenerated.

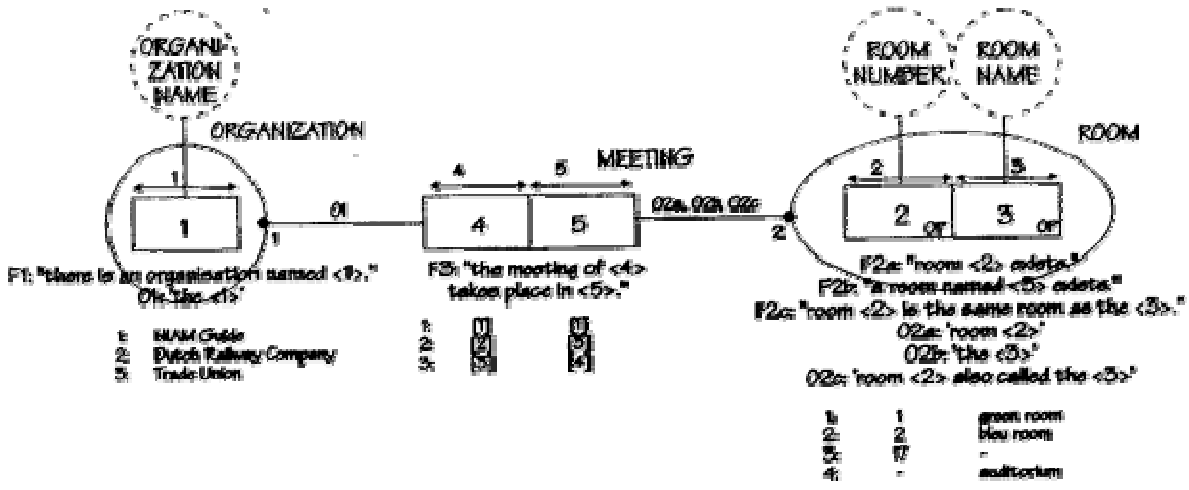


Figure 4.1: IG with generalization

In the nominalized fact type ROOM roles 2 and 3 are optional (non-mandatory), which is marked with 'OP' in the IG (see also § 3). The hyphens represent null values in the sense of (De Troyer, 1993, pp. 18-22), that is in the "no information' interpretation.

The nominalized fact type ROOM violates the traditional NIAM rule stating that nominalized fact types must have exactly one uniqueness constraint covering all its roles (so-called *n-rulefor nominalization).* In FCO-NIAM this well-formedness requirement (a meta-grammatical constraint) on IG's is attenuated, reading:

> *Every role in a non-lexical object type (i.e. a nominalized facttype) must be covered by at least one uniqueness constraint,. every combination of roles that is covered by a certain uniqueness constraint must have its own existence postulating fact type expression and its own object type expression.*

Of course, for a nominalized fact type only those tupel populations are allowed for which an object type expression is recorded. This is implied by the communication and substitution principles (P1 and P5 in § 1.1).

A less conceptually(or if you wish: more practically) inclined information analyst would probably have introduced a unique room number for each room. In that case the IG shown in figure 4.2 would have originated. We challenge the reader to try to add FTE's, OTE's and LP's to this IG in such a way that the original fact expressions can be regenerated from the IG plus its LP.

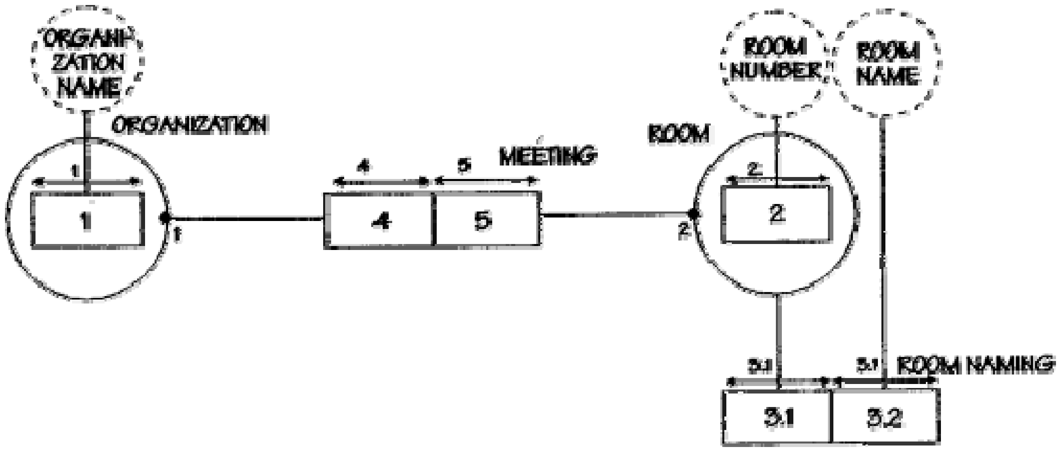


Figure 4.2: IG with artificial room number

A final remark:

- From an FCO-NIAM IG without complex identification structures, the GLR-algorithm discussed in §3 generates a relational schema which obeys the so-called 'entity integrity requirement' (every table must have a primary key being a combination of mandatory roles covered by a uniqueness constraint). If generalization occurs in an IG, then the algorithm leaves the multiple identification just as it was in the initial IG. Although in our experience some database administrators don't mind multiple identification, it is not difficult to avoid this by introducing a unique identifier or by splitting multiply identified tables. This option can easily be incorporated in our CASE tool, but at the moment this has not yet been done, because we want to incorporate all consequences of complex identification (see also paragraphs 4.2 and 4.3) in a single update of our CASE tool, which is now on hand.

4.2 The need for recursive identification structures

The example discussed in this paragraph is taken from a project carried out by the engineering bureau of the Dutch Railway Company. Co-author Van der Lek was involved as a senior information analyst. Among other things all information on so-called OS-sheets had to be modeled. OS stands for Overview Signals. Figure 4.3 shows a fragment of such an OS-sheet, a bit stylized, together with a legend. We have emphasized a few lines by drawing them bold.
Using the legend it is not difficult to verbalize a few facts:

"Signal <u>54</u> exists."
"Signal <u>46</u> exists."
...
"A route can be set up between <u>signal 54</u> and <u>signal 46</u>."
"A route can he set up between <u>signal 46</u> and <u>signal 20</u>."

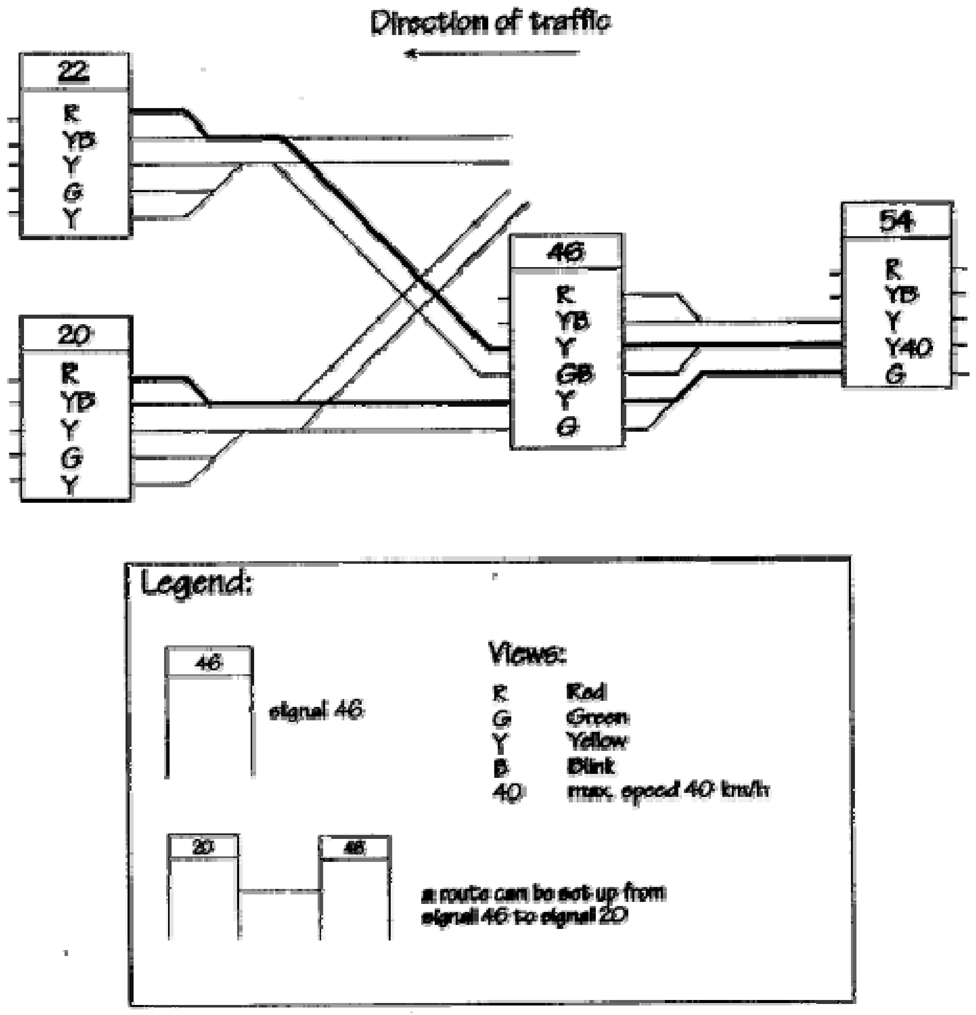


Figure 4.3: Fragment of OS-sheet

The corresponding. fact types SIGNAL and ROUTE are drawn in figure 4.4. From figure 4.3 it is not quite clear which views a signal might possibly show. On being asked the user declared that every signal may show every possible view in principle. So:

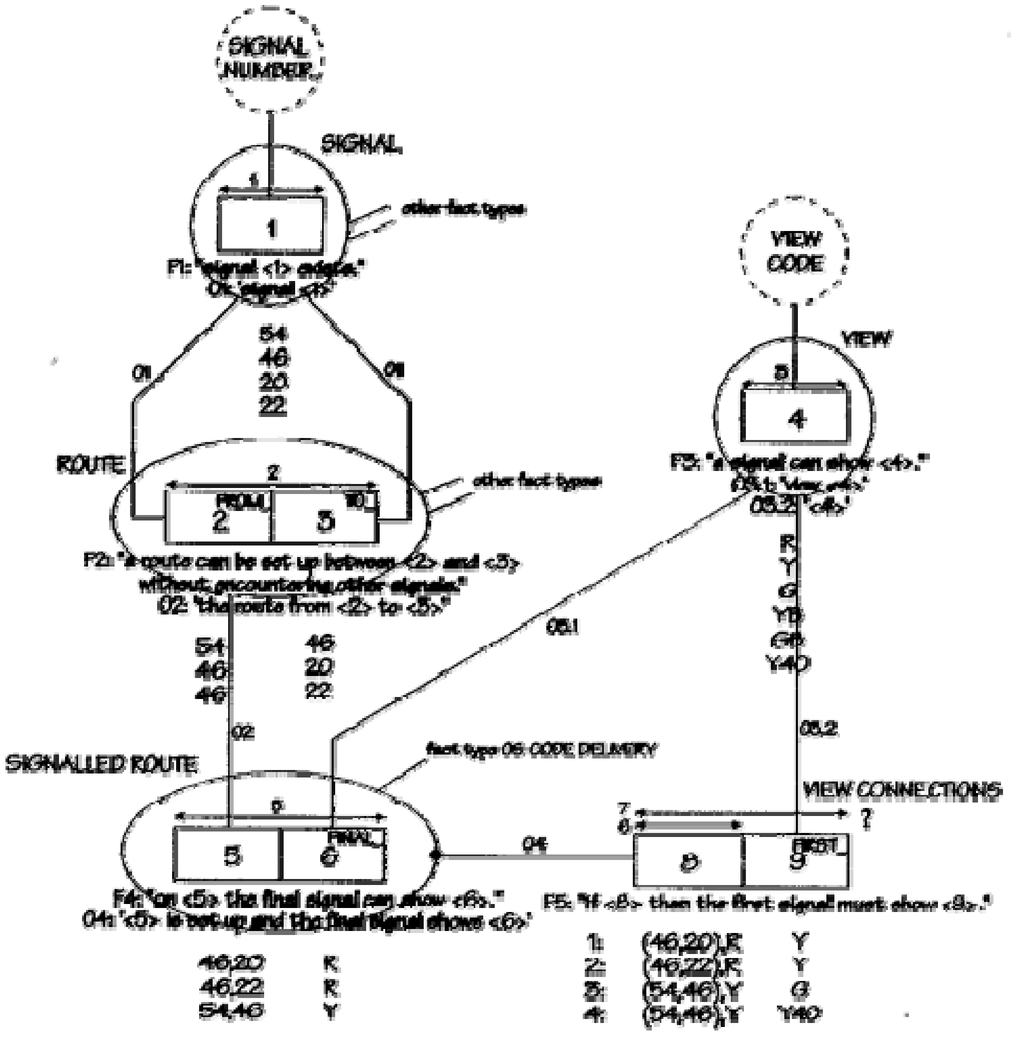
> "A signal can show <u>R</u>."
> "A signal can show <u>Y</u>."
> ...
> "A signal can show <u>GB</u>."
> "A signal can show <u>Y40</u>."

See fact type VIEW in figure 4.4. Obviously not all these possible views are relevant for a certain signal on the OS-sheet. Only the views actually used by a signal are drawn in. This became quite clear when we investigated the meaning of the connection lines between signals. The two bold lines between signals 20 and 46 and between signals 22 and 46 were verbalized as follows:

> 1: "If the route from signal 46 to signal 20 is set up and the final signal shows view R then the first signal must show Y."
> 2: "If the route from signal 46 to signal 22 is set up and the final signal shows view R then the first signal must show Y."

An experienced FCO-NIAM analyst knows of course that sentences in the form of "If ... then ... ." rules can, in good NIAM fashion, simply be regarded as fact expressions. The resulting fact type is called VIEW CONNECTIONS. It has FTE: "If < ROUTE > and the final signal shows <VIEW> then the first signal must show < VIEW >." It is however not drawn as a ternary fact type in figure 4.4 as might be expected, because later on another fact type had to be added as well: OS_CODE_DELIVERY, resulting from sentences like:

> "If the route from signal 46 to signal 20 is set up and the final signal shows view R then the OS-code X must be delivered to the train."



SIGNAL NUMBER
SIGNAL
other fact types
F1: "signal <1> exists."
O1: 'signal <1>'
54
46
20
22
O1
O1
VIEW CODE
VIEW
F3: "a signal can show <4>."
O3.1: 'view <4>'
O3.2: '<4>'
ROUTE
FROM
TO
other fact types
F2: "a route can be set up between <2> and <3> without encountering other signals."
O2: "the route from <2> to <3>."
R
Y
G
YB
GB
Y40
54
46
46
46
20
22
O3.1
O2
O3.2
fact type OS CODE DELIVERY
SIGNALLED ROUTE
VIEW CONNECTIONS
FINAL
O4
FIRST
?
F4: "on <5> the final signal can show <6>."
O4: '<5> is set up and the final signal shows <6>'
F5: "If <8> then the first signal must show <9>."
46,20 R
46,22 R
54,46 Y
1: (46,20),R Y
2: (46,22),R Y
3: (54,46),Y G
4: (54,46),Y Y40

Figure 4.4: View connections, wrong modeling

Sentences belonging to VIEW CONNECTIONS and OS_CODE_DELIVERY have the sentence part between 'If'...... 'then' in common. The NIAM procedure then requires to create a separate nominalized fact type SIGNALED ROUTE providing a common object type expression in order to avoid redundancy. After further classification and qualification the information grammar in figure 4.4 was completed and populated with the facts put into words so far. (For easy readability we use labels instead of tupel references.)

At this point a problem arises. Fact typeexpression F5 in figure 4.4: "If <8> then the first signal must show <9> ." strongly suggests a functional dependency between objects of object type SIGNALED ROUTE and objects of object type VIEW. Therefore in fact type VIEW CONNECTIONS uniqueness consul 6 was imposed on role 8 only. But after we added all the facts concerning the other two bold lines in figure 4.3 (a tupel in ROUTE, a tupel in SIGNALED ROUTE with role values '(54,46)' and 'Y', and tupels 3 and 4 in VIEW CONNECTIONS), uniqueness constraint 6 had to be replaced by uniqueness constraint 7 because of the common role value of role 8 in tupels 3 and 4. As a check we regenerated the fact expressions for tupels 3 and 4, yielding the following two sentences:

> 3: "If the route from signal 54 to signal 46 is set up and the final signal shows view Y then the first signal must show G."
> 4: "If the route from signal 54 to signal 46 is set up and the final signal shows view Y then the first signal must show Y40."

Obviously this is wrong. Uniqueness constraint 6 was the right one of course and in order to ensure that it would not he violated the view Y had to be replaced by two new virtual views Y-1 and Y-2. See figures 4.5 and 4.6.

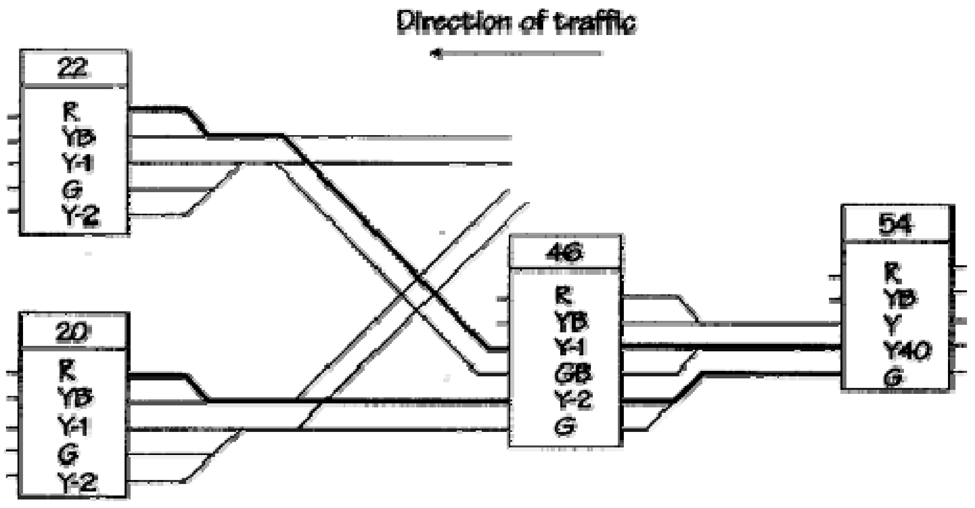


Figure 4.5: OS-sheet with virtual views

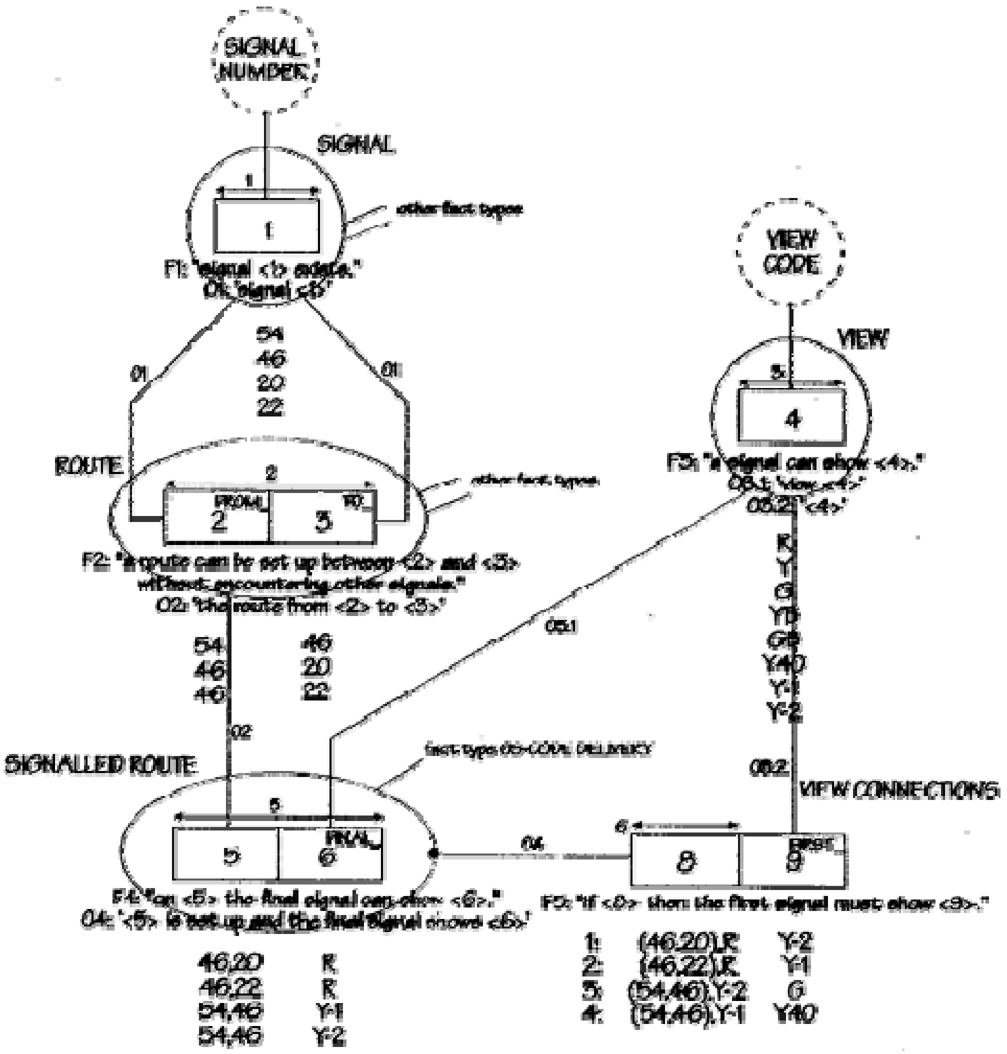
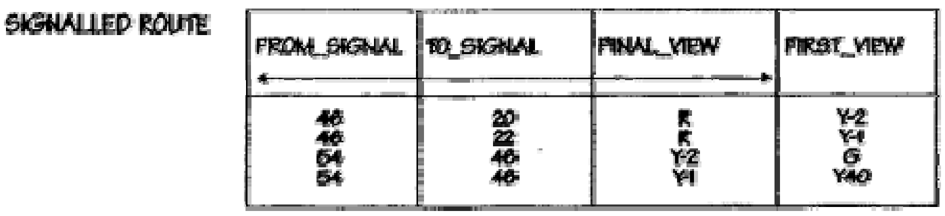


Figure 4.6: IG with virtual views

This solves the problem. Tupels 1 to 4 now represent the following sentences:

1: "If the route from signal 46 to signal 20 is set up and the final signal shows view R then the first signal must showY-2."
2: "If the route from signal 46 to signal 22 is set up and the final signal shows view R then the first signal must show Y-1."
3: "If the route from signal 54 to signal 46 is set up and the final signal shows view Y-2 then the first signal must show G."
4: "If the route from signal 54 to signal 46 is set up and the final signal shows view Y-1 then the first signal must show Y40."

In a relational database the table SIGNALED ROUTE (figure 4.7) was created. This table is operational today for benchmarking purposes.


SIGNALLED ROUTE

FROM_SIGNAL	TO_SIGNAL	FINAL_VIEW	FIRST_VIEW
46	20	R	Y-2
46	22	R	Y-1
54	46	Y-2	G
54	46	Y-1	Y40

Figure 4.7: Relational table with virtual views

Although the database manager was quite satisfied with this solution the designers of the OS-sheets didn't like this at all. Why were they forced to introduce these in their opinion irrelevant view codes Y-1 and Y-2 (and so on?) in their OS-sheets? They had never needed them before. From the point of view of the 100 percent conceptual principle (P2 in § 1.1) they were absolutely right! After a new interview session with the designers of the OS-sheets we rephrased the four bold connections in figure 4.3 in the following way:

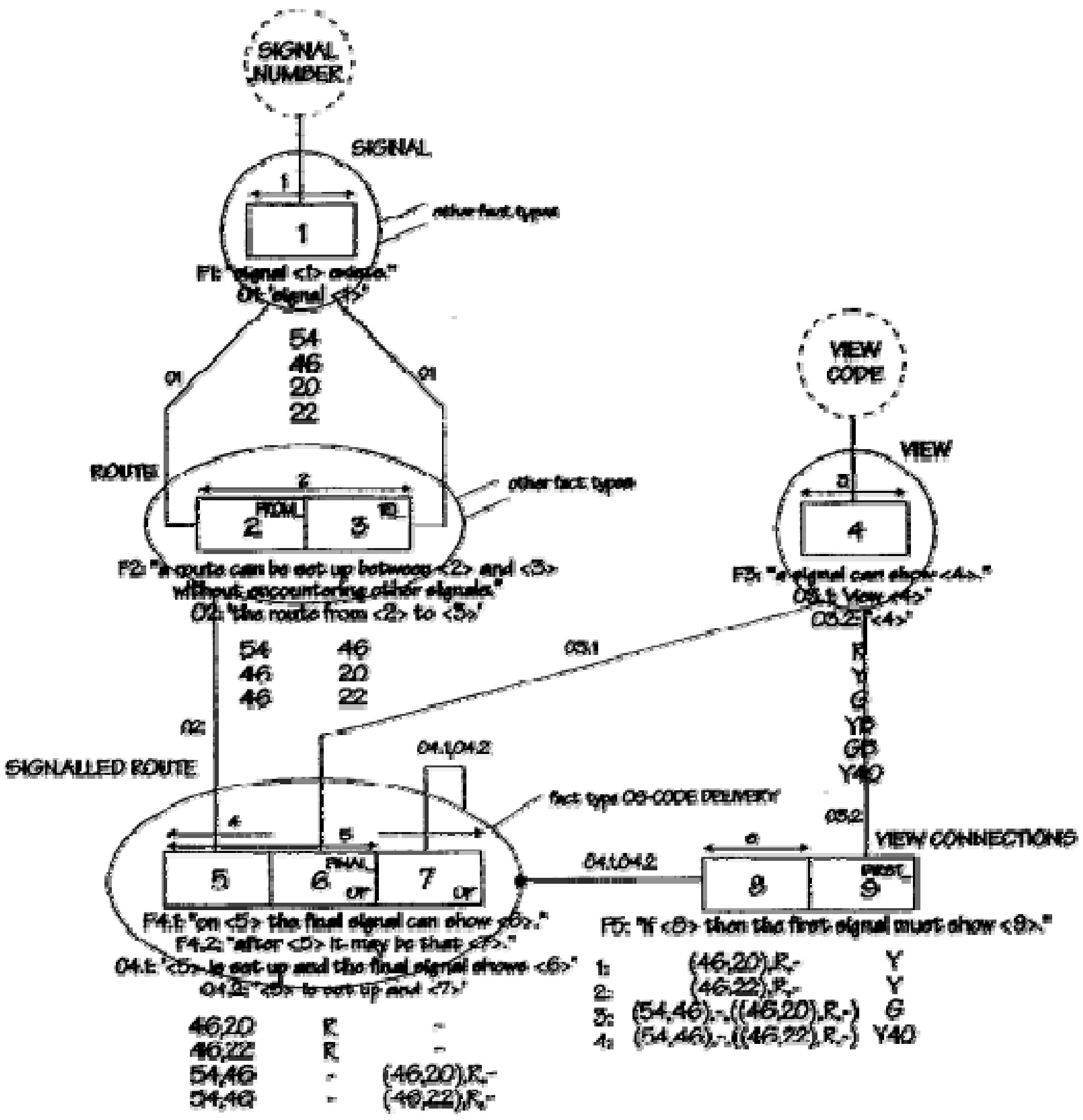
> 1: "If the route from signal 46 to signal 20 is set up and the final signal shows view R then the first signal must show Y."
> 2: "If the route from signal 46 to signal 22 is set up and the final signal shows view R then the first signal must show Y."
> 3: "If the route from signal 54 to signal 46 is set up and the route from signal 46 to signal 20 is set up and the final signal shows view R then the first signal must show G."
> 4: "If the route from signal 54 to signal 46 is set up and the route from signal 46 to signal 22 is set up and the final signal shows view R then the first signal must show Y4O."

Although in figure 4.3 only one-step and two-step view connections occur, it is quite obvious that potentially no limitation on the number of routes chained into one signaled exists. We conclude: object type SIGNALED ROUTE has a recursive identification structure. The way we model this in FCO-NIAM satisfies the conceptualization principe and is shown in figure 4.8. It can he easily checked that for object type SIGNALED ROUTE the n-rule for nominalization (see § 4. 1) holds.

Now we can regenerate the fact expressions 1 to 4 from the populated IG in figure 4.8 via the substitution principle. In order to be able to regenerate recursive fact expressions we incorporated recursive identification in FCO-NIAM. In fact, the example discussed in this paragraph was one of the first real life cases where we really needed it. For another striking example see (Van der Lek, 1993 A; 1993 B). Other researchers treat recursion as well (Ter Hofstede, 1993), however without incorporating the substitution principle.

A few final remarks:

- We have not introduced any new concepts at the metagrammar level, because the generic principle (P6, § 1.1) already implies the necessity to allow non-mandatory roles as population of our GenMG. See § 3 where the GLR-algorithm generates such roles/columns. This means that our present CASE-tool can already store information grammars like figure 4.8. Only the model-specific constraints defining a well-formed elementary FCO-NIAM IG had to be slightly slackened.
- We are presently working on a refinement of the GLR-algorithm so we can have automatical virtual number generation replaying recursive identification structures. This is necessary as long as the Relational Model does not support recursivity.
- If we would use PROLOG or a knowledge system shell instead of a relational database management system for the implementation of IG's, then we could realize such recursive identification structures straightaway. In any case we can now deal with complex rules having a recursive structure in a pure conceptual way.



SIGNAL NUMBER
SIGNAL
other fact types
F1: "signal <1> exists."
O1: 'signal <1>'
54
46
20
22
O1
O1
VIEW CODE
VIEW
ROUTE
other fact types
FROM
TO
F2: "a route can be set up between <2> and <3> without encountering other signals."
O2: 'the route from <2> to <3>'
F3: "a signal can show <4>."
O3.1: 'view <4>'
O3.2: '<4>'
54 46
46 20
46 22
O3.1
R
Y
G
YR
GB
Y40
O2
O4.1,O4.2
SIGNALLED ROUTE
fact type OS-CODE DELIVERY
O3.2
VIEW CONNECTIONS
FINAL
FIRST
O4.1,O4.2
F4.1: "on <5> the final signal can show <6>."
F4.2: "after <5> it may be that <7>."
O4.1: '<5> is set up and the final signal shows <6>'
O4.2: '<5> is set up and <7>'
F5: "if <8> then the first signal must show <9>."
1: (46,20),R,- Y
2: (46,22),R,- Y
3: (54,46),-,((46,20),R,-) G
4: (54,46),-,((46,22),R,-) Y40
46,20 R -
46,22 R -
54,46 - (46,20),R,-
54,46 - (46,22),R,-

Figure 4.8: IG with recursive identification structure

4.3 The need for sequence and set based identification

In this paragraph we describe an example taken from (Van der Lek, 1993 B), which played an important part in discussions held in The Netherlands.

Quamsplashing is a (fictitious) game. Two teams having an arbitrary number of members are pitted against each other. The rules of this enjoyable game are irrelevant for our purposes. An example schedule for a quamsplash contest is shown in figure 4.9.

GAME SCHEDULE FOR TONIGHT'S QUAMSPLASH CONTEST TEAMS:	
Steve, Colin	Macy, Marty, John
Ted, Marty, Cindy	Colin, Steve
Marty	Steve, Cindy, John, Ted

Figure 4.9: Quamsplash schedule

A verbalization of these facts (all of fact type GAME SCHEDULE) by the contest officials (object expressions have been designated by double underlining already):

"The team consisting of Steva and Colin plays against the team consisting of Macy, Marty and Jogh."

"The team consisting of Ted, Marty and Cindy plays against the team consisting of Colin and Steve."

"The team consisting of Marty plays against the team consisting of Steve, Cindy, John and Ted."

We verified that it is also possible to phrase the second sentence as:

"The team consisting of Ted, Marty and Cindy plays against the team consisting of Steve and Colin."

The information analyst concludes from these fact expressions, that the fact type GAME SCHEDULE is a homogeneous binary fact type, in which both roles are played by the non-lexical object type TEAM. The fact type expression for GAME SCHEDULE is:

" < TEAM > plays against < TEAM >."

The object type expression for object type TEAM is:

"the team consisting of < PERSON > [[, < PERSON >...] and <PERSON>]'.

We use the Backus Naur Form (BNF) notation here. Square brackets ('[' and ']') denote optional parts; the three dots ('...') mean that the part ', <PERSON>' may be repeated ad libitum. The number of repetitions is of course always final at the instance level, although it is potentially infinite at the type level.

Further analysis (again accomplished by classification and qualification) of object expressions like 'the team consisting of Macy, Martin and John', yields an object type expression '<NAME>' for the non-lexical objects of the object type PERSON, in which NAME is an identifying label type for object type PERSON.

Most Probably a practically inclined information analyst would solve the identification problem for the non-lexical object type TEAM by introducing a new label type TEAM NUMBER. He would then assign a different team number to each of the 5 (not: 6!) different teams and add some appropriate fact type expressions and object type expressions. The populated information grammar would then he the one depicted in figure 4.10.

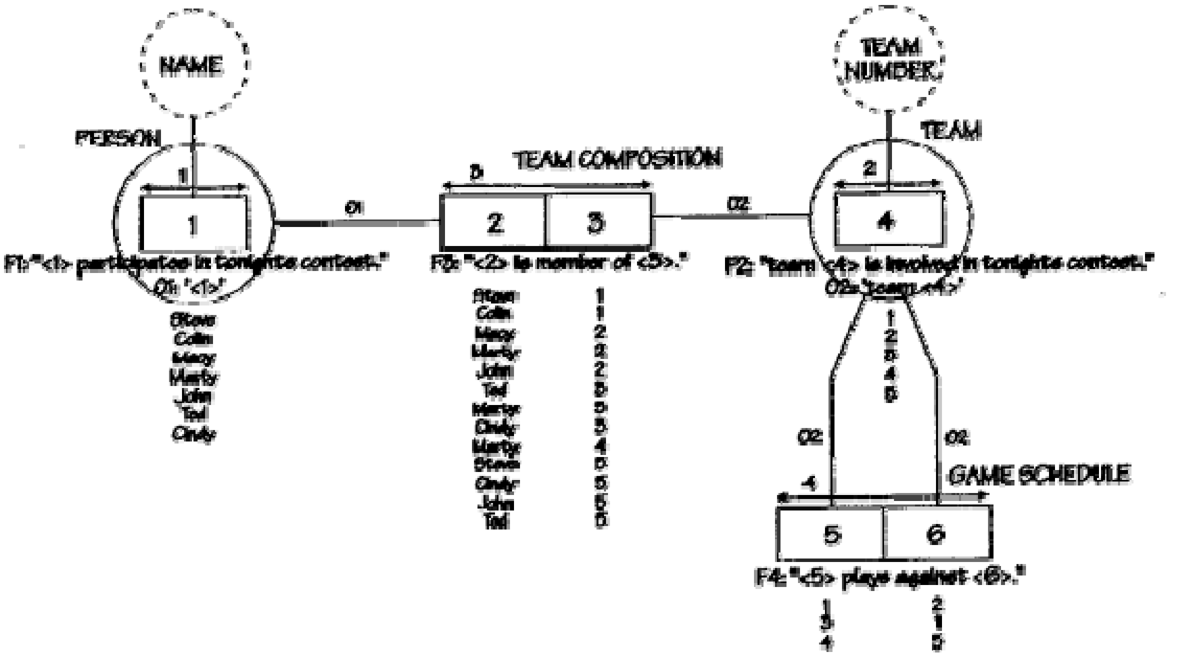


Figure 4.10: IG with artificial label type TEAM NUMBER

But if the contest officials wouldn't need these artificial team numbers at all ("We always did it this way; we just write down the names of the team members and everybody knows what to do.'), then something is wrong in this approach. Indeed, the 100 percent conceptualization principle is violated again. The information analyst might carry this through anyway (and would probably introduce other artificial numbers identifying persons or games), but that does not concern us now.

If we want to comply with the 100 percent conceptualization principle (P2, § 1.1) then we must at least be capable of modeling such identification structures. It was accomplished by the Dutch researcher Ter Hofstede by introducing a set type (Ter Hofstede, 1993, pp. 25-31). However, his solution doesn't comprise the modeling of sentence type and object type expressions. Figure 4.11 shows an FCO-NIAM IG modeling game schedules for quamsplash contests, in which object type TEAM is modeled as a set type.

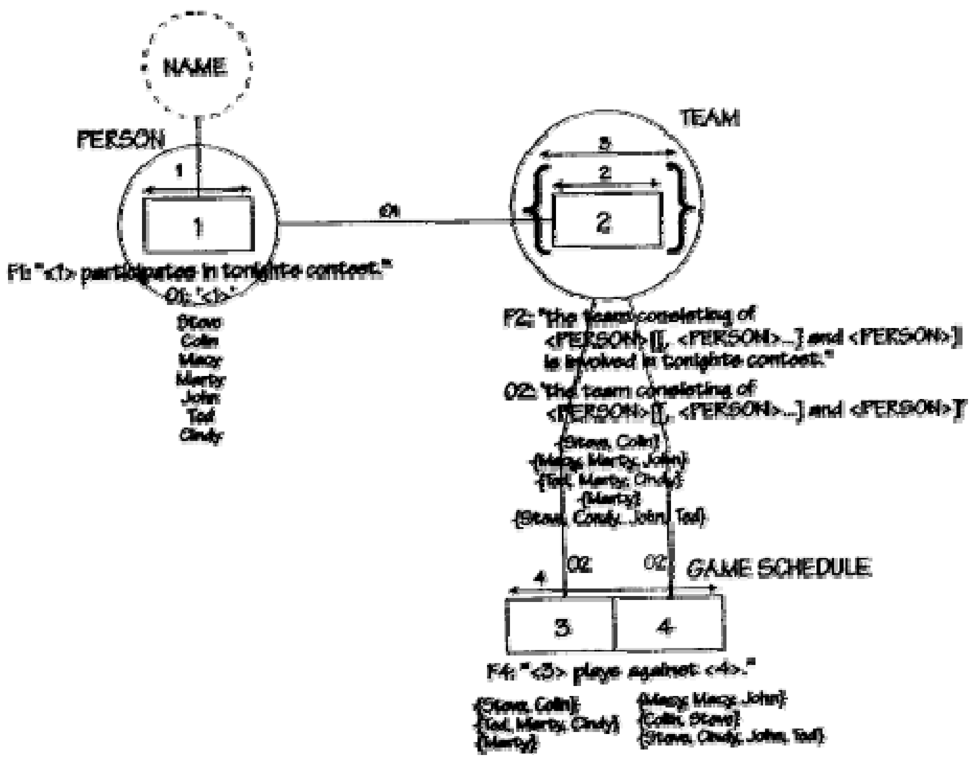

NAME
PERSON
1
1
O1
TEAM
3
2
2
F1: "<1> participates in tonight's contest."
O1: '<1>'
Steve
Colin
Macy
Marty
John
Ted
Cindy
F2: "the team consisting of <PERSON>[[, <PERSON>...] and <PERSON>] is involved in tonight's contest."
O2: "the team consisting of <PERSON>[[, <PERSON>...] and <PERSON>]"
{Steve, Colin}
{Macy, Marty, John}
{Ted, Marty, Cindy}
{Marty}
{Steve, Cindy, John, Ted}
O2
O2
GAME SCHEDULE
4
3
4
F4: "<3> plays against <4>."
{Steve, Colin}
{Ted, Marty, Cindy}
{Marty}
{Macy, Macy, John}
{Colin, Steve}
{Steve, Cindy, John, Ted}

Figure 4.11 : IG with set type

The braces '{' and around role 2 designate a set type. An element of a set may occur in it only once. Uniqueness constraint 2 implies this. Uniqueness constraint 3 enforces that each set uniquely defines a team. It is not difficult to regenerate the original verbalizations from the IG plus the three tupels of GAME SCHEDULE via the substitution process.

A few final remarks:

- In the case of multi-sets we drop the uniqueness constraint within the braces. The same holds for sequence types which we designate by parentheses, as is usual in mathematics.
- Ter Hofstede also showed in his PhD-thesis that the introduction of set types and sequence types has far-reaching consequences for meta-modeling and consequently also for the conceptual architecture of CASE tools (Ter Hofstede, 1993, pp. 130-134). Doing so, he gave us an important research item for the future development of our FCO-NIAM based CASE tool. For instance: in the IG's presented so far we identify uniqueness constraints by a code, whereas identification by the collection of participating roles is also possible. So our FCO-NIAM meta grammar would be more elegant from a conceptual point of view if it used set types itself. The main reason why IG's like

figure 4.11 cannot be recorded in our FCO-NIAM-based CASE-tool at the moment is our wish to preserve this conceptual elegance; it is not because there are any problems with the recording of variable length fact types, fact type expressions or object type expressions or with the assimilation of the consequences for substitution.
- The consequences of introducing set types and sequence types for the GLR algorithm are not trivial.
- If we would use PROLOG or a knowledge system shell instead of a relational database management system for the implementation of IG's, then we could realize such set or sequence identification structures straightaway. In any case we can now deal with complex rules having a set or sequence structure in a pure conceptual way.

5. Specialization

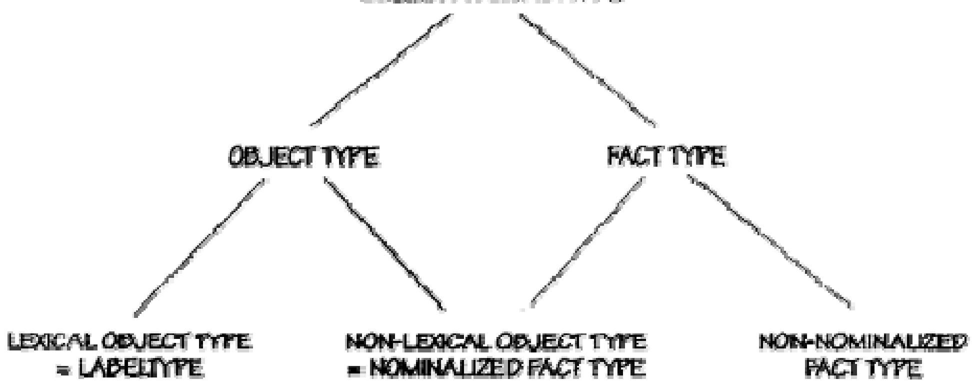
Like generalization, we can incorporate specialization in FCO-NIAM without introducing new concepts at the meta-grammar level, as most others do (Nijssen & Halpin, 1989; Ter Hofstede, 1993, pp. 34-36). At least in the structural sense the whole thing can be done in terms of the concepts introduced in the preceding paragraphs. We illustrate this with an example in the context of meta-modeling. This paragraph therefore has two purposes:

- to illustrate the way we deal with subtypes in FCO-NIAM.
- to clarify the structure of the GenMG.

5.1 Specialization or subtyping

In (fully) communication oriented information modeling we unified non-lexical object types and fact types into a single concept, which incidentally is the only populatable construct. Besides this there was of course the concept lexical Object type (label type). Label types are the ultimate sources of role values with which the populatable constructs are actually populated. In (Van der Lek, Bakema & Zwart, 1992; Van der Lek, 1993 A) we used the term 'Objecttype' as a generic term for all these concepts, but we now prefer to use the term objecttype/facttype', because this is a better reflection of its function of unifying all constructs which can either play roles (i.e. lexical and non-lexical object types) or which can contain roles (fact types, nominalized or not).

The subtype structure of the role containing constructs and the role playing constructs in our GenMG is shown in figure 5. 1. This can of course be checked by the familiar subtype matrix method for determining subtype hierarchies. The corresponding part of the GenMG in FCO-NIAM form is depicted in figure 5.2.


OBJECTTYPE/FACTTYPE
OBJECT TYPE
FACT TYPE
LEXICAL OBJECT TYPE = LABELTYPE
NON-LEXICAL OBJECT TYPE = NOMINALIZED FACT TYPE
NON-NOMINALIZED FACT TYPE
Figure 5.1: Subtype structure of OBJECTTYPE/FACTTYPE

The non-lexical object type OBJECTTYPE/FACTTYPE is the pater familias of the subtype hierarchy; it is a nominalization of its existence postulating unary fact type with the same name together with its fact type expression and object type expressions (Remark: NAME is a labeltype prefixed here by 'OT/FT_'; 'or' is used in the non-exclusive sense.):

FTE1: "There is an object type or fact type < OT/FT_NAME >."
OTE1.1: 'the objecttype or facttype < OT/FT_NAME >'
OTE1.2: '< OT/FT_NAME >'

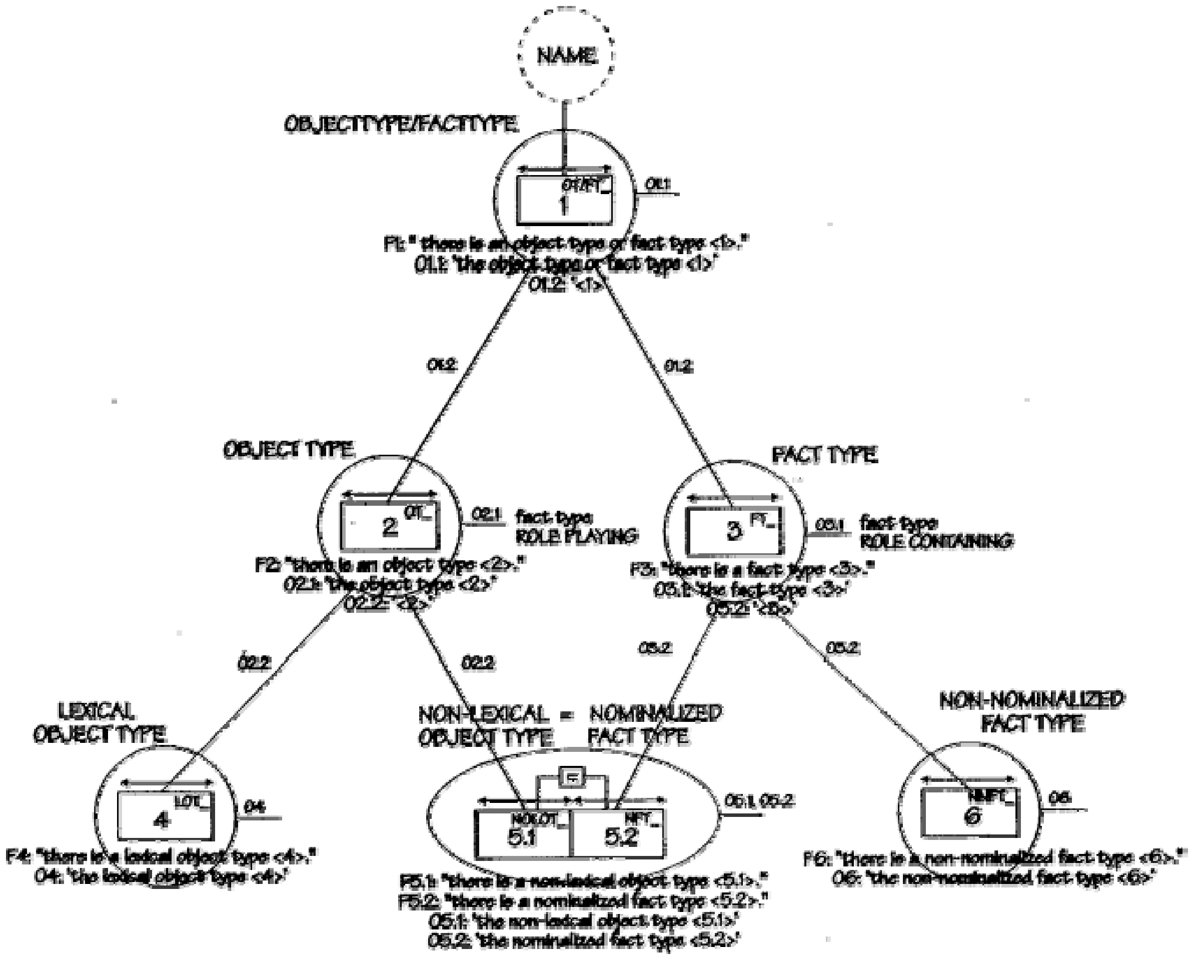

NAME
OBJECTTYPE/FACTTYPE
OT/FT_
1
O1.1
F1: " there is an object type or fact type <1>."
O1.1: 'the object type or fact type <1>'
O1.2: '<1>'
O1.2
O1.2
OBJECT TYPE
2
OT_
O2.1
fact type ROLE PLAYING
F2: "there is an object type <2>."
O2.1: 'the object type <2>'
O2.2: '<2>'
FACT TYPE
3
FT_
O3.1
fact type ROLE CONTAINING
F3: "there is a fact type <3>."
O3.1: 'the fact type <3>'
O3.2: '<3>'
O2.2
O2.2
O3.2
O3.2
LEXICAL OBJECT TYPE
LOT_
4
O4
F4: "there is a lexical object type <4>."
O4: 'the lexical object type <4>'
NON-LEXICAL OBJECT TYPE = NOMINALIZED FACT TYPE
=
NOLOT_
5.1
NFT_
5.2
O5.1, O5.2
F5.1: "there is a non-lexical object type <5.1>."
F5.2: "there is a nominalized fact type <5.2>."
O5.1: 'the non-lexical object type <5.1>'
O5.2: 'the nominalized fact type <5.2>'
NON-NOMINALIZED FACT TYPE
NNFT_
6
O6
F6: "there is a non-nominalized fact type <6>."
O6: 'the non-nominalized fact type <6>'

Figure 5.2: Subtype structure in GenMG of OBJECTTYPE/FACTTYPE

In the first sublayer we have the non-disjunct subtypes of the pater familias OBJECT TYPE and FACT TYPE. In the second layer we have the disjunct subtypes LEXICAL OBJECT TYPE and NON-LEXICAL OBJECTTYPE (NOMINALIZED FACT TYPE) of the supertype OBJECTTYPE and the subtypes NOMINALIZED FACT TYPE (NON -LEXICAL OBJECT TYPE) and NOT-NOMINALIZED FACT TYPE of the supertype FACT TYPE. The second layer is a partition (total and exclusive) of the supertype OBJECTTYPE/FACTTYPE, whereas the first layer is a total but non-exclusive division of the pater familias.

All object instances of the object type OBJECT TYPE play a role in the binary fact type ROLE_PLAYING between OBJECT TYPE and ROLE (not shown) having as FTE: "<OBJECT TYPE> plays <ROLE>." All object instances of object type FACTTYPE play roles in other fact types as well, such as the binary facttype ROLE_CONTAINING between FACTTYPE and ROLE (not shown) having as FTE: " < FACTTYPE > contains <ROLE>." All subtypes in the second layer as well as the pater familias OBJECTTYPE/FACTTYPE also play roles in other fact types (not shown either).

Please note that subtype OBJECT TYPE in the first layer is modeled as a nominalization of its existentence postulating unary fact type OBJECT TYPE in which the pater familias OBJECTTYPE/FACTTYPE plays the single role. The same holds for subtype FACT TYPE. The identification structure for the common subtype NON-LEXICAL OBJECT TYPE (i.e. NOMINALIZED FACT TYPE) of OBJECT TYPE and FACT TYPE is perhaps somewhat surprising, but its multiple identification is already familiar from § 4.1. In fact this illustrates a beautiful interrelationship between generalization and specialization: we can regard NON-LEXICAL OBJECT TYPE both as a subtype in the whole hierarchical subtype structure and as a generalization of its direct parents OBJECT TYPE and FACT TYPE. The substitution principle holds of course as can be checked easily.

A few final remarks:

- The question of inheritance arises immediately. We will not elaborate on this question here, but only remark that we have a fact type in the GenMG for the explicit recording of which fact types have the inheritance property. We leave this important discussion for a future paper in which we will present the whole GenMG.
- In figure 5.2 there is a small rectangular box connected to roles 5.1 and 5.2 with '=' in it. This is our graphical representation of what we call a *strict equality constraint.* Such a constraint imposes the equality of role values within each tupel. This supplements ordinary equality constraints already defined in NIAM, which are usually depicted with '=' in a little circle. The need for strict equality constraints was already encountered many times in the practice of NIAM information modeling. We also introduced a strict exclusion constraint. Both strict versions can, like the non-strict ones, also be used in inter fact type situations and must then be interpreted in the 'inner-equi-join' sense.

5.2 Derivable and declarative subtypes

There are two ways of specifying which subpopulations of an supertype are also populations of its various subtypes. We call this subtype specification. In NIAM the classical way to specify subtypes is to require that for each subtype a so-called 'subtype defining rule' must be given, in which:

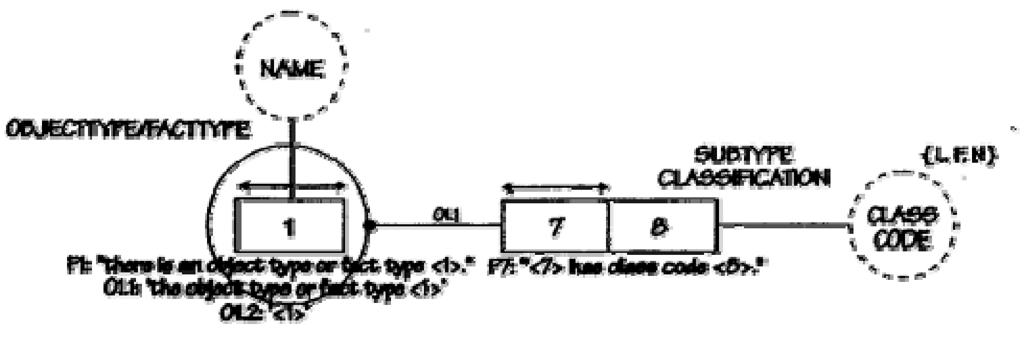
> *A subtype specification is formulated in terms of fact types having roles played by super type(s) of the subtype being specified.*

In FCO-NIAM this way of subtype specification may be used as well as can be seen in figure 5.3. However, in FCO-NIAM subtype specification rules are ordinary derivation rules. This means that the subtypes in figure 5.2 must be marked with an asterisk (denoting derivable fact types) if we use this kind of subtype specification.

A slightly different version of derivable suptype specification is shown in figure 5.4, in which we use a more sophisticated existence postulating fact type for the pater familias OBJECTTYPE/FACTTYPE. A new way to specify subtypes is that any subtype may also have a definition, in which:

> *A subtype specification is formulated in terms of existence postulating fact type(s) of supertypes.*

So: in FCO-NIAM a subtype hierarchy is not necessarily modeled using only derivable facttypes. It can also be done in a pure declarative style. In that case the existence postulating fact types of the different subtypes are not derivable from other fact types, but bear the burden of the subtype specification themselves. In this case figure 5.2 doesn't need any derivation rulesadded because it already contains all the necessary information.


NAME
OBJECTTYPE/FACTTYPE
SUBTYPE CLASSIFICATION
{L,F,N}
1
OL1
7
8
CLASS CODE
F1: "there is an object type or fact type <1>."
OL1: "the object type or fact type <1>"
OL2: "<1>"
F7: "<7> has class code <8>."

A tuple x of OBJECTTYPE/FACTTYPE is a tuple of

OBJECT TYPE if
"the object type or fact type x has class code L." or
"the object type or fact type x has class code N."

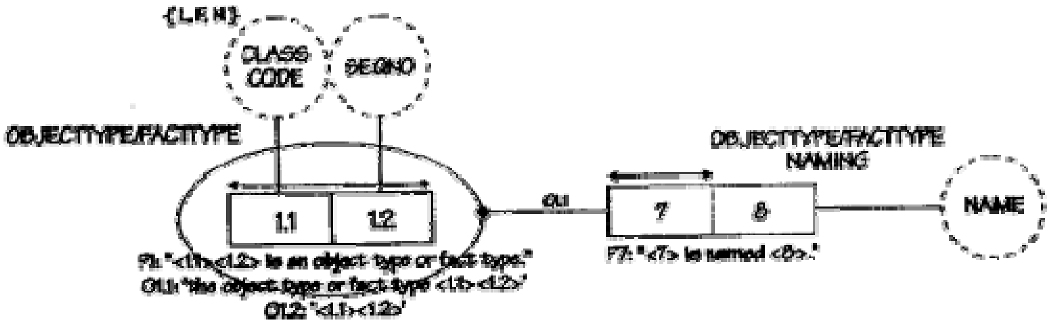
FACT TYPE if
"the object type or fact type x has class code F." or
"the object type or fact type x has class code N."

LEXICAL OBJECT TYPE if
"the object type or fact type x has class code L."

NON-LEXICAL OBJECT TYPE (NOMINALIZED FACT TYPE) if
"the object type or fact type x has class code N."

NON-NOMINALIZED FACT TYPE if
"the fact type x has class code F."

Figure 5.3: Subtype specification: derivable


{L,F,N}
CLASS CODE
SEQNO
OBJECTTYPE/FACTTYPE
OBJECTTYPE/FACTTYPE NAMING
1.1
1.2
OL1
7
8
NAME
F1: "<1.1><1.2> is an object type or fact type."
OL1: "the object type or fact type <1.1><1.2>"
OL2: "<1.1><1.2>"
F7: "<7> is named <8>."

All tuples with a role value

L or N in role 1.1 are tuples of OBJECT TYPE
F or N in role 1.1 are tuples of FACT TYPE
L in role 1.1 are tuples of LEXICAL OBJECT TYPE
N in role 1.1 are tuples of NON-LEXICAL OBJECT TYPE (NOMINALIZED FACT TYPE)
F in role 1.1 are tuples of NON-NOMINALIZED FACT TYPE

Figure 5.4: Subtype specification: declarative

Final remarks:

- We will skip the point of dereferencing role values in the subtype specifying derivation rules in figure 5.3.

- Derivable and declarative subtype specifications can both be used in one subtype hierarchy.
- The GLR-algorithm implemented in our CASE-tool collapses all derivable subtypes with their supertypes before the grouping step is carried out. Surprisingly this can be done by the grouping process described in § 3.3, as the reader can check for himself. But he must be warned for a little snake in the grass: he will encounter a nice recursive structure, which does no harm at all if we take into account that strict equality roles are redundant in relational schemata.
- The GLR-algorithm does not affect declarative subtypes, because we would loose information otherwise.

6. The educational and practical impact of FCO-NIAM

The extension of NIAM presented in this paper yielding Fully Communication Oriented NIAM is accepted widely today in the Dutch NIAM-world. It is being used by several software houses for information modeling. The aspects of FCO-NIAM based on principles Pl to P6 as presented in our previous paper (Van der Lek, Bakema & Zwart, 1992) were adopted in (Nijssen & Schouten, 1993; Nijssen, 1993). Several college- and university-level institutes in The Netherlands use these course books and our own course materials (m which the main accent lies on principles P5 and P6 and the consequences thereof for CASE-architecture (Bakema, Zwart & students, 1993)) in progams of Computer Science, Business Informatics and Information System Development. Other course materials developed very recently (Willemsen, 1994) will be used next course year by several college-level institutes.

- The same holds for the HG CASE-tool our students built (HG stands for Hogeschool Gelderland). This CASE-tool is developed to serve as a didactical aid and is fully transparent in the sense that it can be used to demonstrate:
- the relational version of the GenMG discussed in § 3.2, which in reality is somewhat more sophisticated in some aspects. This relational version of the GenMG can of course also be generated from an FCO-NIAM version by the implemented GLR algorithm. As a consequence the HG CASE-tool can generate its own relational meta schema and in doing so it supports the didactical process.
- the step by step processing of the GLR algorithm (§§ 3.3, 4.1, 5.2).
- automatic generation of documentation (both textual and graphical) for each fact type or relational tupel, in all phases of the GLR process.
- automatic SQL-based database generation (DDL-statements, including constraint definition).
- how a Data Dictionary can be generated automatically in the form of descriptions of the structure and integrity aspects of the generated database plus (very sophisticated for a data dictionary) fact type expressions for the information stored in the database.

These features, the last certainly not being the least, turned out to be quite interesting for commercial companies as well. The meta grammar used in our CASE-tool has been nominated as the internal standard by software house CVI (Centre For Information, Utrecht, The Netherlands).

This year our students built new versions of our CASE-tool for the Dutch software house BSO/Management Support Baarn and for several college-level institutes. In these new versions they implemented a sentence interface as illustrated in figure 2.2, which works for multiple identification structures (see § 4.1) as well. In that case generalized object types are created automatically.

Acknowledgements

The graphical symbol for set types in figure 4. 1 1 was proposed by Kriss Yunker in one of the inspiring discussions we had with him. He will certainly pardon us for having extended this symbol with a set based existence postulating fact type expression and object type expression.

We thank Susanne Willemsen for reading the text of this paper, suggesting some improvements to the first draft version and designing all the figures. She is one of our students and will graduate in july 1994 on a project developing course materials about Fully Communication Oriented information modeling (Willemsen, 1994).

About the authors

Drs. Guido Bakema and drs. Jan Pieter Zwart are both computer science professors associated with the Hogeschool Gelderland in Arnhem and Nijmegen in the Netherlands. They develop and teach courses about information system development and CASE architecture, not only for the college-level degree program in Computer Science and System Development, but also for training programs for fellow computer science professors on the college- and university-level and for software house employees on the senior-level. They are members of 'the NIAM Group', a Dutch group of people from the business, university and educational world discussing formal, conceptual, educational and practical aspects of fact-oriented information system specification.

Dr. Harm van der Lek works as a senior advisor for the Dutch software house BSO/Management Support in Baarn and is specialized in information analysis in complex domain, and conceptual problems. He introduced NIAM as the methodological standard in the Dutch Railway Company and several other companies. He is chairman of 'the NIAM Group'. He advised the Hogeschool Gelderland and several other educational institutes to base their teaching programs about information system development on FCO-NIAM and he inspired Hogeschool Gelderland to build an FCO-NIAM oriented CASE-tool for didactical purposes.

Bibliography

- Bakema, G.P., J.P.C. Zwart & students (1993), Communicatiegeoriënteerde informatiemodellering en CASE-architectuur, internal coursebook, Hogeschool Gelderland, Nijmegen (the Netherlands)
- De Troyer, O.M.F. (1993), On data schema transformation, PhD thesis Tilburg (The Netherlands), ISBN 90-900591-3-X
- Griethuysen, J.J. (1982) editor, Concepts and terminology for the conceptual schema and the information base, Publ. nr. ISO/TC97/SCS-N695
- Nijssen, G.M. & T.A. Halpin (1989), Conceptual schema and relational database design, Prentice-Hall. Englewood Cliffs (New Yersey)
- Nijssen, G.M. & H. Schouten (1993), Materiaal voor bedrijfseconomie en bedrijfskunde, PNA Publishing bv, Beutenaken (The Netherlands)
- Nijssen, G.M. (1993), Universele Informatiekunde, PBA Publishing bv, Beutenaken (The Netherlands)
- Ter Hofstede, A.H.M. (1993), Information modelling in data intensive domains, PhD thesis Nijmegen (the Netherlands), ISBN 90-9006236-X
- Van der Lek, H., G.P. Bakema & J.P.C. Zwart (1992), De unificatie van objecttypen en feittypen, een praktisch en didactisch vruchtbare theorie, Informatie (may 1992), volume 34, nr. 5, pp. 279-295
- Van der Lek, H. (1993 A), On the formal structure information grammar, Proceedings of the 1993 NIAM gUIde working conference at Utrecht (the Netherlands)
- Van der Lek, H. (1993 B), Natuurlijke taal Informatie Analyse Methode: een overzicht, Handboek Database Systemen, pp. 5143-1-5143-37, Array Publications, Alphen a/d Rijn (The Netherlands)
- Van der Lek, H. (1993 C), Het groeperingsalgoritme in een communicatiegeoriënteerde aanpak, internal paper, Hogeschool Gelderland, Nijmegen (The Netherlands)
- Willemsen, S.H. (1994), Volledig communicatiegeoriënteerde informatiemodellering, masters thesis, Hogeschool Gelderland, Nijmegen (The Netherlands)