Universität Karlsruhe
Fakultät für Informatik
76128 Karlsruhe, Germany

Translating E/R-diagrams into Consistent Database Specifications

Thomas Fuchß
Institut für Logik, Komplexität und Deduktionssysteme
Universität Karlsruhe
76128 Karlsruhe, Germany
email: fuchss@ira.uka.de

Interner Bericht Nr. 2/94

Translating E/R-diagrams into Consistent Database Specifications

Thomas Fuchß[1]

January 28, 1994

[1]This research was sponsored by the BMFT project Korso.

Abstract

In this report we present the results of a case study with the KIV system (Karlsruhe Interactive Verifier). This case study deals with consistency proofs for entity relationship database specifications. In comparison to other case studies done with the KIV system (e.g. [Sch89, Ste93]) a completely different task was tackled by the KIV approach. Not the development of correct programs were to the fore, but the examination of the suitability of the module concept as a basis for consistency proofs of specifications. The realization of this task consists of the development of a modular system as the target of the translation and a realistic example for testing the method.

Preface

The following report is part of the central case study HDMS-A([CFLW92])[1] within the German national project KORSO[2]. This study is dedicated to the development of a complex information system for the support of the patient data administration in the specialized heart disease clinic DHZB[3]. While the developers group PMI[4] develops the real system for the clinic called HDMS, the project KORSO's aim was the rigorous development of an abstracted version of HDMS by exclusive use of pure formal methods. The abstraction refers both to number of modeled documents and depth of treatment, while still considering the relevant aspects in a partly parameterized way.

The investigation of HDMS-A has been done by 11 partners within KORSO. An extended overview provides [CHL94a, CHL94b]. The main topics are: an actual state analysis of the selected documents in the patient record as well as of the existing and motivating problems concerning safety and security, distribution and effectiveness (see [CKL93]); the requirements analysis and specification, beginning with a description of the chosen policy ([HHM+93, Huß93]) and two technical formalisms providing means for the translation of entity-relationship diagrams as well as data flow diagrams into SPECTRUM (see [Het93, Nic93]), finally the requirements specification itself ([HHM+93]); the main field of safety and security treated and in general ([GH93]) and concretely: [Ste93, Ren94]; finally the investigation of the integration of existing software components into a formal development: [Con93a, Con93b, Dam93, BS93, Shi94, MZ94]. Apart from those there were few specialized contributions to selected topics: [Hec93, Aut93, Ben93, HS93].

Any of the cited reports can be obtained directly from the authors (see [CHL94b] for the procedure).

---

[1] The management system HDMS: Heterogeneous Distributed Information Management System is the successor of the PADCOM-System developed at the Deutschen Herzzentrum Berlin during the BERKOM-Project.
The postscript A stands for abstract.

[2] Korrekte (=correct) Software, sponsored by the German ministry for research and technology.

[3] Deutsches Herzzentrum Berlin.

[4] Projektgruppe Medizin Informatik am DHZB und der TU Berlin.

Contents

Chapter 1

Introduction

For more and more areas in software engineering the application of formal methods becomes of interest. E.g., such areas include fields where human life is at risk or the fields where data protection is necessary. In the field of data protection formal methods have a special place. Only formal specifications permits to guarantee a special behavior of a system by formal verification.

To make this claim treatable it is necessary to establish a bridge between semi-formal specification methods which are often used in the area of database development and formal specifications necessary for formal verification. The problem, however, is to prove the consistency of the resulting specification. For the realization of this problem we use a trick: The translation not only generates a structured specification but also a structured implementation for all types and operations. So the consistency proof can be reduced to a program verification problem. As an example for this method we regard the semi-formal specification method *entity relationship diagrams*.

In the database community entity relationships are well known structures for modeling connections between real world objects (for example see [tB92]). Entity relationship diagrams (E/R-diagrams) contain entities as objects of consideration. These objects are described through attributes. Each attribute has a sort called attribute sort. Among these attributes some are marked as mandatory. Each entity contains at least one mandatory attribute as distinguishing feature (key). Entities with the same attributes are collected in entity types. Between the entity types in an E/R-diagram binary relations exist. We distinguish four kinds of relations: (1 to 1), (1 to n), (n to 1) and (m to n). As an example to test our developed method for entity relationships we present *Cardiac-Catheterisation*, part of the HDMS-A E/R-diagram *Treatment* (see [HHM+93]).

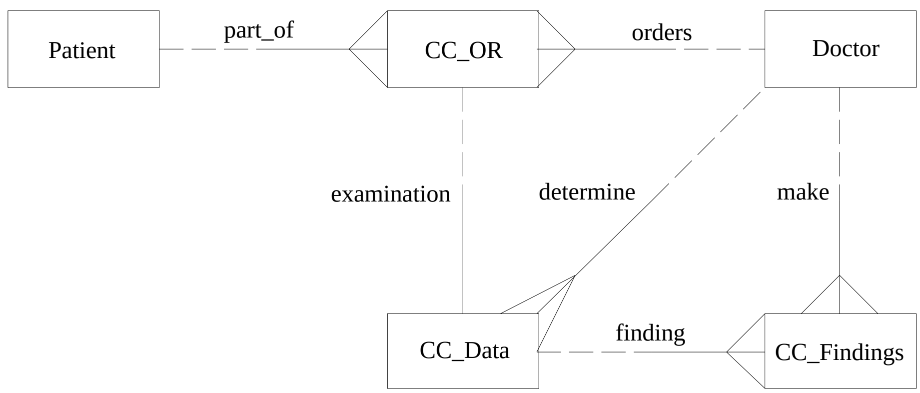
Example 1.0.1 (Cardiac-Catheterisation)

The following figure shows a typical diagram for an entity relationship. It includes five entity types Patient, CC_OR *(cardiac catheter order),* CC_Data, CC_Findings *(the conclusions drawn from the data) and* Doctor *as well as six relationships* part_of, orders, examination, determine, make *and* finding.

The Relationships:

name	**between**		**type**
part_of	*Patient*	*CC_OR*	*1 to n*
orders	*Doctor*	*CC_OR*	*1 to n*
examination	*CC_OR*	*CC_Data*	*1 to 1*
determine	*Doctor*	*CC_Data*	*1 to n*
make	*Doctor*	*CC_Findings*	*1 to n*
finding	*CC_Data*	*CC_Findings*	*1 to n*

The Diagram:



In the following table we present the different entity types with their attributes. Special attributes are market as mandatory *(m) or* mandatory and part of the primary key *(m/k). The sorts of the attributes are omitted.*

Patient	**CC_OR**	**CC_Data**	**CC_Findings**	**Doctor**
Patient_Id (m/k)	*CCOR_Id (m/k)*	*CCOR_Id (m/k)*	*Findings_Id (m/k)*	*Doctor_Id (m/k)*
Name (m)	*makingDate (m)*	*CCR*	*FindingsData (m)*	*Name (m)*
Sex (m)	*Comment*	*ExaminationDate (m)*	*Findings (m)*	*Address (m)*
Birthdate (m)		*Start (m)*	*Report*	*Rank (m)*
Birthplace (m)		*End*		*Ward*
CostBearer (m)		*PressureCurve*		*Entry (m)*
Address		*x-ray-film*		*Leaving*
FamDoctor				*Role*
physicalData				
Ward				
Room				

To model such an E/R-diagram for example by a relational database our Munich KORSO partners described a translation for E/R-diagrams into a SPECTRUM specification to give them a formal algebraic semantic (see [Het93]). Entity types are specified as sets of tuples of attributes with an additional key function and the relationships as sets of pairs.

In contrary to SPECTRUM ([BFG$^+$93]), the specification language currently used in the KIV system is mainly first order sorted predicate logic. This means especially that all functions are strict and total. The semantics of KIV specifications is loose. It is possible to restrict possible models to term generated ones by the clause **generated by** (or **freely generated by**). Structured specifications are supported. The different types of specifications and structuring operations are explained where appropriate. It is not the intention of this report to describe in detail the features of the KIV system (specification language, programming language, logic, calculus, methodology etc.). The reader is referred to the theoretical papers and technical documentation of the KIV system [HRS87], [HRS89], [HRS90] or [Rei92].

The KIV system currently requires all symbols to be unique, resulting in a large number of similar symbols. This handicap can be partly overcome by a uniform naming and renaming process (e.g. using the sort or an abbreviation of the sort as an index).

There is a clear distinction between programs and specifications. The programming language is PASCAL-like, and contains the empty statement **skip**, the never terminating statement **abort**, assignments $x := \tau$, conditionals **if** ε **then** α **else** β, compound statements $\alpha;\beta$, local variables **var** $x = \tau$ **in** α, procedure declarations, while statements and procedure calls $p(\tau_1,\ldots,\tau_n;x_1,\ldots,x_m)$ where $\tau_1,\ldots,\tau_n$ are the value parameters and $x_1,\ldots,x_m$ the var-parameters.

Properties of programs are expressed in dynamic logic (DL, see [HRS89]). A diamond formula $\langle\alpha\rangle\,\varphi$ where α is a program and φ again a DL formula, states that α terminates and afterwards φ holds. A sequent calculus is used to prove the correctness of such formulas.

Because of these KIV features — especially the first order specification language — we have used the SPECTRUM specification of the E/R-database as a loose pattern for developing our translation mechanism.

1.1 The Method For Proving Consistency

If we want to describe the semantic of structures like E/R-diagrams by transforming them into algebraic specifications, it is necessary to guarantee that the resulting specification is consistent.

To carry out the consistency proof for the database specification resulting by our translation method we used the module concept of the KIV approach for the development of correct large software systems (see [Rei92]).

A module is a triple consisting of an export specification, an implementation and an import specification, describing the data procedures operate upon. The implementation is a collection of such procedures and a mapping. The mapping sets up the correspondence between the sorts and operations of the export specification and the sorts and procedures of the import. The semantic of a module is a partial function induced by the implementation and the mapping which maps a generated model of the import specifications to a generated model of the export signature. A module is correct or short the implementation is correct if and only if the semantic function is total and all results fulfill the export axioms. The conditions for correctness of the module can be uniformly expressed through properties over the implementation which are formulated in Dynamic Logic (also see [Rei92]).

This makes a proof of consistency for a specification possible by creating an implementation of the specification over a consistent import specification using correctness preserving structuring operations for the specifications.

Because the database specification is uniform for every E/R-diagram we translate arbitrary E/R-diagrams via a PPL program[1] into a modular system containing structured specifications and implementations. An advantage of this method is, that we obtain a prototype-like implementation of the resulting database specification. Furthermore, for every concrete E/R-diagram it is possible to prove the consistency explicitly with the KIV system. A generalized proof for arbitrary but fixed E/R-diagrams is also possible and sketched in this paper.

In the following chapters we will concentrate on the elements of the modular system and how they are combined to the algebraic specification of a relational database for an arbitrary E/R-diagram. Furthermore we present the concrete instantiation of the schemes for the example *Cardiac-Catheterisation* (see above).

Apart from this pure consistency check it is possible to enrich the database specification for an E/R-diagram with transactions. This means we have to add new top level functions to the pure database specification, which represent the functionality of the transactions to an user. These functions can be implemented on the basis of the database specification and it is possible to formulate and prove integrity conditions for the transactions. We are working on doing so for the example *Cardiac-Catheterisation*. The results of this work will be presented in a forthcoming report.

[1]PPL: Proof Programming Language, the meta language of KIV, a typed functional language ML-like, in which all our tactics are implemented.

In chapter 2 we present an overview of the modular system which we used for the consistency proof. In chapter 3 we present the details of the modular system which are independent of the E/R-diagram. Chapter 4 present the core of the translation, the generated specifications and implementations which are dependent of the used E/R-diagram.

Finally, I wish to thank my KIV colleagues Wolfgang Reif, Gerhard Schellhorn, Arno Schönegge and Kurt Stenzel for many valuable discussions, and our student Wolfgang Ahrendt for doing the implementation of the transformation algorithm and his part by developing the specification.

Chapter 2

The Modular System

First we give an overview of the modular system which we use for the proof of consistency. Details will be stated more precisely in the following chapters.

The modular system which is generated by the translation of an E/R-diagram is schematically shown in figure 2.1, where the arrows represent the used-by-relation between specifications. Solid arrows from one specification to another represent the fact that the specification pointed to is used in the other specification. Dashed arrows point to generic specifications where the specification at the basis of the arrow is an actualized version by instantiating the parameter with the specification at the solid arrow-head. The triple of rectangular boxes and boxes with rounded corners represent modules with export specification (above), implementation pointed to by ragged arrows and import specification (below).

The modular system can be separated in two parts:

The static part: Consists of specifications and modules which are used for arbitrary E/R-diagrams. They represent fixed data structures which are used in each translated database specification. They all stem from a library.

Specifications for modeling relations:

- elem: A specification of a parameter — a sort with an order.
- elem1: A renamed version of *elem.*
- elem2: A renamed version of *elem.*
- pair: A generic specification with parameter *elem1 + elem2* (the union of both).
- ordered-pair: An enrichment of *pair* with a lexical order over pairs.
- orderset: A generic specification with parameter *elem.*
- pair-orderset: An actualization of ordered-set with ordered-pair.

Specifications for modeling sets of entities:

- elem-with-key: A specification of a parameter.
- coded-set: A generic specification with parameter *elem-with-key.*

Modules:

- ordered-pair–pair: The implementation of the enrichment.
- The implementation of orderset by a list specification.
- The implementation of coded-set by another list specification.

These implementations and the *list* specifications are not presented in this report, because they are of no interest for the translation and all stem from a library.

The generated part: The core of the translation. Dependent on a given E/R-diagram the following specifications and modules are generated automatically:

Specifications:

- AttributesParam: A parameter specification, with orders for some attributes.

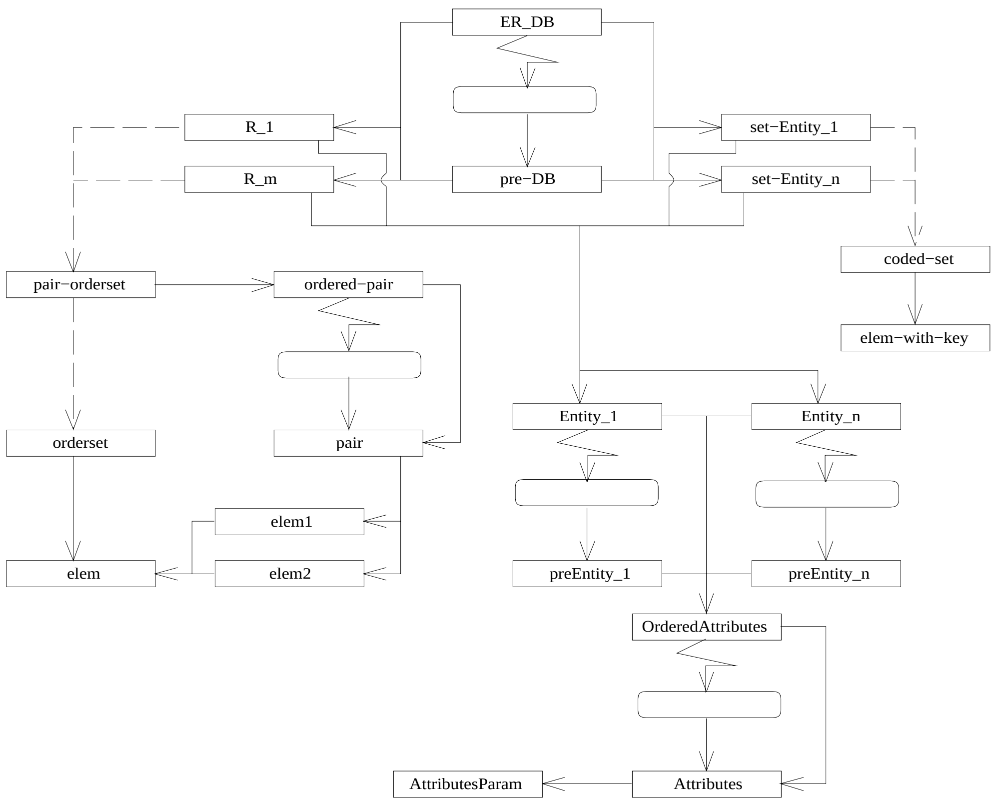


Figure 2.1: Modular System

- Attributes: A generic specification with parameter *AttributesParam*, The specification introduces new constants *error* and *undef* for each sort in the specification *AttributeParam*.
- OrderedAttributes: An enrichment of *Attributes* with axioms which enlarge the orders to the new constants.
- preEntity(1–n): A specification using *OrderedAttributes*. A consistent basis for the implementation of the specification *Entity*
- Entity(1–n): An enrichment of *OrderedAttributes* with a sort E, a sort *keysort-E* and corresponding functions and axioms. One specification for each entity type.
- set-Entity(1–n): An actualization of *coded-set* with *Entity(1–n)*, for modeling entity types.
- R(1–n): An actualization of *pair-orderset* with the union of two specifications *Entity*, for modeling the relation ships.
- pre-DB: A specification using *set-Entity* and *R*. A consistent basis for the implementation of the specification *ER_DB*
- ER_DB: An enrichment of *set-Entity*$_1$ $+\ldots+$ *set-Entity*$_n$ $+R_1+\ldots+R_m$ with the sort *db* and corresponding functions and axioms, which describe the database.

The specifications *Attributes, preEntity*$_1$ to *preEntity*$_n$ and the specification *Pre-DB* are all of a special form which guarantees consistency with respect to the parameter and used specifications.

Modules:

- OrderedAttributes–Attributes: An implementation of the enrichment.
- Entity–preEntity(1–n): An implementation of *Entity* using *preEntity*.
- ER_DB–pre-DB: An implementation of *ER_DB* using *pre-DB*.

Chapter 3

The Static Part

In this chapter we concentrate our view to the static parts of the modular system, those modules and specifications which are the same for each E/R-diagram. These Specifications represent the basic structures for modeling the relational database.

3.1 The Specification coded-set

The *coded-set* specification is used actualized in the database specification for modeling entity types. In the original SPECTRUM specification no further properties are required for the set specification. For our plan an ordinary set specification is not suitable.

If we want to implement the database specification on the basis of a specification using the same inputs (see chapter 4) we must have possibilities for selecting elements in sets. This is only possible if we have a set of ordered elements. To avoid such an order of entities we introduce an order of the keys of the elements and guarantee that the keys of two elements in one set are pairwise different. This is justifiable because in every real database an order of the keys is necessary for an efficient implementation. Assuming this *key property* makes an implementation in the module *ER_DB–pre-DB* unnecessary.

The specification *elem-with-key* is the parameter of the *coded-set* specification. An element sort with an error element and an accompanying key sort, an encode function and order axioms over the key sort is specified. The error element is necessary if we want to select in the set *error-c-set* (the error element of *coded-set*, see below) or if we select with a key and there is no corresponding element in the set.

elem-with-key =
specification
  sorts element, key;
  constants error-elem : element;
  functions encode : element $\to$ key ;
  predicates . $\ll_{key}$. : key $\times$ key;
  variables an-elem: element; a-key$_2$, a-key$_1$, a-key: key;
  axioms
    $\neg$ a-key $\ll_{key}$ a-key,
    a-key $\ll_{key}$ a-key$_1$ $\wedge$ a-key$_1$ $\ll_{key}$ a-key$_2$ $\to$ a-key $\ll_{key}$ a-key$_2$,
    a-key $\ll_{key}$ a-key$_1$ $\vee$ a-key $=$ a-key$_1$ $\vee$ a-key$_1$ $\ll_{key}$ a-key
end specification

The specification can be assumed as consistent, only an order is specified over the key sort. Remarkable is that the encode function needs not to be injective, since some entities may have the same key. These entities represent the "same object" in different coinage, e.g. findings before and after writing the report.

coded-set =
generic specification
parameter elem-with-key **target**
sorts coded-set;
constants empty-c-set : coded-set; error-c-set : coded-set;
functions

. $+_{cs}$.	:	coded-set × element	→	coded-set	**prio** -5 **left**;
. $-_{cs}$.	:	coded-set × element	→	coded-set	**prio** -5 **left**;

selector functions:
selecting with a key

sel	:	key × coded-set	→	element	;

selecting the element with the minimal key

min-cs	:	coded-set	→	element	;

the rest after deleting the element with the minimal key

rest-cs	:	coded-set	→	coded-set	;

predicates
. in-cs . : element × coded-set;
a well founded order used for induction instead of a size function
. realsub-cs . : coded-set × coded-set;
variables el_2, el_1, el: element; cs_2, cs_1, cs: coded-set;
axioms
coded-set **generated by** empty-c-set, $+_{cs}$, error-c-set;
empty-c-set ≠ error-c-set,
error-c-set $-_{cs}$ el = error-c-set,
cs $-_{cs}$ error-elem = error-c-set,
¬ error-elem in-cs cs,
¬ el in-cs error-c-set,
min-cs(empty-c-set) = error-elem,
min-cs(error-c-set) = error-elem,
rest-cs(error-c-set) = error-c-set,
¬ error-c-set realsub-cs cs,
¬ cs realsub-cs error-c-set,

key property
el = error-elem
∨ cs = error-c-set
∨ ¬ el in-cs cs ∧ sel(encode(el), cs) ≠ error-elem
↔ cs $+_{cs}$ el = error-c-set,

el ≠ error-elem → empty-c-set $-_{cs}$ el = empty-c-set,
cs $+_{cs}$ el ≠ error-c-set → cs $+_{cs}$ el $-_{cs}$ el = cs $-_{cs}$ el,
encode(el_1) ≠ encode(el_2) → cs $+_{cs}$ el_1 $-_{cs}$ el_2 = cs $-_{cs}$ el_2 $+_{cs}$ el_1),
¬ el in-cs empty-c-set,
cs $+_{cs}$ el_2 ≠ error-c-set → (el_1 in-cs cs $+_{cs}$ el_2 ↔ el_1 = el_2 ∨ el_1 in-cs cs),
cs_1 ≠ error-c-set ∧ cs_2 ≠ error-c-set
→ (cs_1 = cs_2 ↔ (∀ el.el in-cs cs_1 ↔ el in-cs cs_2)),
el in-cs cs → sel(encode(el), cs) = el,
(∀ el.el in-cs cs → encode(el) ≠ a-key) ↔ sel(a-key, cs) = error-elem,
cs ≠ empty-c-set ∧ cs ≠ error-c-set → rest-cs(cs) $+_{cs}$ min-cs(cs) = cs,
rest-cs(empty-c-set) = empty-c-set,
el in-cs rest-cs(cs) → encode(min-cs(cs)) $\ll_{key}$ encode(el),
cs_1 ≠ error-c-set ∧ cs_2 ≠ error-c-set
→ cs_1 realsub-cs cs_2 ↔ cs_1 ≠ cs_2 ∧ (∀ el.el in-cs cs_1 → el in-cs cs_2)
end generic specification

To guarantee the consistency of the specification *coded-set* an implementation on the basis of a list specification was done. Because only the *coded-set* specification is of interest we have omitted the list specification and the implementation.

3.2 The Specification pair-orderset

Like the specification *coded-set*, the *pair-orderset* specification is used actualized in the database specification. It is necessary for modeling the binary relations between entities. This modeling is uniform for arbitrary relations because the different kinds of relations (1 to 1), (1 to n), (n to 1) and (m to n) are of interest only in static integrity conditions. But integrity conditions only make sense for transactions and not for the primitive operations, which are specified in the database specification *ER_DB* (see below).

For the same reasons as above a structure which consists only of sets of tuples is not suitable for our aim. So we must have possibilities to select pairs in sets.

Like above we can avoid an order over entities because only the keys are used for representing the relations.

3.2.1 The Structure of pair-orderset

The elem Specification

The specification *elem* is the parameter specification used in the specification *orderset* and renamed in the specification *pair*.

elem =
specification
  sorts elem;
  predicates . $\ll_{elem}$. : elem × elem;
  variables e-$_2$, e-$_1$, e: elem;
  axioms
    ¬ e $\ll_{elem}$ e,
    e $\ll_{elem}$ e-$_1$ ∧ e-$_1$ $\ll_{elem}$ e-$_2$ → e $\ll_{elem}$ e-$_2$,
    e $\ll_{elem}$ e-$_1$ ∨ e = e-$_1$ ∨ e-$_1$ $\ll_{elem}$ e
end specification

The specification can be assumed as consistent, only an order is specified over the sort *elem* and no other assumption is made for the elements of the sort *elem*.

The orderset Specification

A set specification with the possibility of selecting the minimal element in a set. This is possible through the order which is specified in the parameter specification *elem*.

orderset =
generic specification
  parameter elem **target**
  sorts orderset;
  constants empty-o-set : orderset;
  functions
    . $+_s$. : orderset × elem → orderset **prio** -5 **left**;
    . $-_s$. : orderset × elem → orderset **prio** -5 **left**;
    selector functions:
    selecting the minimal element
    min : orderset → elem ;
    the rest after deleting the minimal element
    rest : orderset → orderset ;
  predicates
    . in . : elem × orderset;

a well founded order used for induction instead of a size function
. realsub . : orderset × orderset;
variables ae_2, ae_1, ae: elem; se_2, se_1, se: orderset;
axioms
orderset **generated by** empty-o-set, $+_s$;
empty-o-set $-_s$ ae = empty-o-set,
se $+_s$ ae $-_s$ ae = se $-_s$ ae,
$ae_1 \neq ae_2 \rightarrow$ se $+_s$ ae_1 $-_s$ ae_2 = se $-_s$ ae_2 $+_s$ ae_1,
$\neg$ ae in empty-o-set,
ae_1 in se $+_s$ $ae_2 \leftrightarrow ae_1 = ae_2 \vee ae_1$ in se,
$se_1 = se_2 \leftrightarrow (\forall$ ae.ae in $se_1 \leftrightarrow$ ae in $se_2)$,
se $\neq$ empty-o-set $\rightarrow$ rest(se) $+_s$ min(se) = se,
rest(empty-o-set) = empty-o-set,
$\forall$ ae.ae in rest(se) $\rightarrow$ min(se) $\ll_{elem}$ ae,
se_1 realsub $se_2 \leftrightarrow se_1 \neq se_2 \wedge (\forall$ ae.ae in $se_1 \rightarrow$ ae in $se_2)$
end generic specification

To guarantee the consistency of the specification *orderset* an implementation on the basis of a list specification was done. Because only the *orderset* specification is of interest we have omitted the list specification and the implementation, like above.

The pair Specification

For modeling relations we need sets of ordered tuples. To get a consistent basis for such tuples we used the KIV concept of *generic data specifications* and specified pairs at first without an order.

A generic data specification corresponds to the *data*-construct in SPECTRUM. The sort *pair* is freely generated by *mkpair*. In addition two selector functions were specified *fst* and *snd*. Their application to an element of sort *pair* yields by *fst* the first component and by *snd* the second component of the pair. Consistency is guaranteed assuming the parameter is consistent.

Because only one parameter specification is possible in the KIV specification language we have to form a union of two renamed versions of the specification *elem* before we can use them as parameter.

elem1 =
rename elem **by morphism**
elem $\rightarrow$ elem1, $\ll_{elem} \rightarrow \ll_{elem1}$, e $\rightarrow e_1$, e-$_1 \rightarrow$ e1-$_1$, e-$_2 \rightarrow$ e1-$_2$
end rename

elem2 =
rename elem **by morphism**
elem $\rightarrow$ elem2, $\ll_{elem} \rightarrow \ll_{elem2}$, e $\rightarrow e_2$, e-$_1 \rightarrow$ e2-$_1$, e-$_2 \rightarrow$ e2-$_2$
end rename

elem1+elem2 = elem1 + elem2

pair =
generic data specification
parameter elem1+elem2
pair = mkpair (. .fst : elem1, . .snd : elem2);
variables par_1: elem1; par_2: elem2; p_2, p_1, p: pair;
end generic data specification

The ordered-pair Specification

The specification *ordered-pair* is the specification of the actualization of the elements in *orderset*. An enrichment of the generic data specification *pair* with the order predicate $\ll_{pair}$ and the corresponding axioms. The specified order is a lexical order. Because the enrichment is hierarchically persistent an implementation on the basis of the specification *pair* is possible for guaranteeing the consistency. We have omitted the implementation in this report too.

ordered-pair =
enrich pair **with**
  predicates . $\ll_{pair}$. : pair × pair;
  axioms
    ¬ p $\ll_{pair}$ p,
    p $\ll_{pair}$ p_1 ∧ p_1 $\ll_{pair}$ p_2 → p $\ll_{pair}$ p_2,
    p $\ll_{pair}$ p_1 ∨ p = p_1 ∨ p_1 $\ll_{pair}$ p,
    p_1 $\ll_{pair}$ p_2 ↔ p_1.fst $\ll_{elem1}$ p_2.fst ∨ p_1.fst = p_2.fst ∧ p_1.snd $\ll_{elem2}$ p_2.snd
end enrich

The pair-orderset Specification

Finally we can describe the specification *pair-orderset* as an actualization of the specification *orderset*.

pair-orderset =
actualize orderset **with** ordered-pair **by morphism**
  elem → pair, $\ll_{elem}$ → $\ll_{pair}$, e → p, e-$_1$ → p_1, e-$_2$ → p_2, orderset → pair-oset, empty-o-set → empty-p-set, $+_s$ → $+_{ps}$, -$_s$ → -$_{ps}$, min → min-ps, rest → rest-ps, in → in-ps, realsub → realsub-ps, ae → ap, ae_1 → ap_1, ae_2 → ap_2, se → ps, se_1 → ps_1, se_2 → ps_2
end actualize

Chapter 4

The Generated Part

In this chapter we present the core of the translation, the *generated part*. The generated part can be divided into three subparts:

- the Attribute Part,
- the Entity Part,
- the Database Part.

4.1 The Attributes

The Attributes are the distinguishing features, the characteristics of the different entities. So if we want to describe E/R-diagrams through algebraic specifications we need specifications for each possible attribute which occurs in the description of an entity. To avoid the necessity to give all these attribute specifications we use one generic specification. This means we don't fix the attributes, but only the sorts and some necessary features. For the mandatory attributes which are part of the primary key, we need an order which guarantees the possibility of searching in sets. This is necessary because in the implementation of the database iteration over sets must be modeled (see section 4.3). If we want to get a concrete consistent specification we only have to instantiate the different sorts of the parameter with consistent specifications.

We describe now the specifications in detail.

4.1.1 AttributesParam

The parameter of the whole specified system. It fixes j different attribute sorts (*normal-Attr*). For every mandatory key attribute of the E/R-diagram an order predicate ($\ll_{n-Attr}$) is specified. The appendix normal is necessary since KIV does not yet allow overloading of identifiers and the identifiers $Attr_i$ are used in the final specification *OrderedAttributes.*

AttributesParam =
specification
  sorts
    normal-Attr$_1$,... normal-Attr$_i$,...normal-Attr$_j$;
  predicates
    ⋮
    $. \ll_{n-Attr_i} .$: normal-Attr$_i$ × normal-Attr$_i$;
    ⋮
  variables
    v-n-Attr$_1$: normal-Attr$_1$;
    ⋮
    v-n-Attr$_{i-2}$, v-n-Attr$_{i-1}$, v-n-Attr$_i$: normal-Attr$_i$;

⋮

v-n-$Attr_n$: normal-$Attr_n$;

axioms

⋮

$\neg$ v-n-$Attr_i \ll_{n-Attr_i}$ v-n-$Attr_i$,

v-n-$Attr_i \ll_{n-Attr_i}$ v-n-$Attr_{i-1}$

$\wedge$ v-n-$Attr_{i-1} \ll_{n-Attr_i}$ v-n-$Attr_{i-2}$

$\rightarrow$ v-n-$Attr_i \ll_{n-Attr_i}$ v-n-$Attr_{i-2}$,

v-n-$Attr_i \ll_{n-Attr_i}$ v-n-$Attr_{i-1}$

$\vee$ v-n-$Attr_i$ = v-n-$Attr_{i-1}$

$\vee$ v-n-$Attr_{i-1} \ll_{n-Attr_i}$ v-n-$Attr_i$

⋮

end specification

The specification can be assumed as consistent, only an order is specified for some sorts and no other assumption is made.

4.1.2 Attributes

Because some attributes of an entity can be mandatory or not it is necessary to enrich the sorts of *Attributes* by "undef" constants for modeling the "unknown state" of attributes. A direct conclusion of this is the possibility of entities to be undefined if mandatory attributes are "unknown". In contrary to Spectrum in the KIV specification language all functions must be total. Therefore undefined entities are matched to error constants. So, by an error propagation we also have to enrich the sorts of the attributes with "error" constants too. This can be done in a consistent way by using the feature of a generic data specification.

The sort *Attr* is freely generated by *error-Attr, undef-Attr* and *copy-Attr*. Where *copy-Attr* is a function which lifts the sort *normal-Attr* from the parameter specification *AttributesParam* to the new sort *Attr*. And *get-Attr* is the corresponding selector. The *copy-Attr-prd* is true if and only if the element is generated by a *copy-Attr* term.

Attributes =

generic data specification

parameter AttributesParam

$Attr_1$ = error-$Attr_1$

| undef-$Attr_1$

| copy-$Attr_1$ (get-$Attr_1$: normal-$Attr_1$)

with copy-$Attr_1$-prd ;

⋮

$Attr_i$ = error-$Attr_i$

| undef-$Attr_i$

| copy-$Attr_i$ (get-$Attr_i$: normal-$Attr_i$)

with copy-$Attr_i$-prd ;

⋮

$Attr_j$ = error-$Attr_j$

| undef-$Attr_j$

| copy-$Attr_j$ (get-$Attr_j$: normal-$Attr_j$)

with copy-$Attr_j$-prd ;

variables

v-$Attr_1$: $Attr_1$;

⋮

v-$Attr_{i-2}$, v-$Attr_{i-1}$, v-$Attr_i$: $Attr_i$;

⋮

v-Attr$_j$: Attr$_j$;
end generic data specification

4.1.3 OrderedAttributes

For all sorts which are mandatory the order predicates have to be expanded for the two new constants. This was done by enriching the generic data specification *Attributes* by new predicates $\ll_{Attr}$ and axioms. In this way we get the specification *OrderedAttributes*. To make the generation more simple the variables which are necessary to formulate these axioms are defined in the specification *Attributes*.

OrderedAttributes =
enrich Attributes **with**
predicates
⋮
$. \ll_{Attr_i} . \quad : \quad$ Attr$_i$ × Attr$_i$;
⋮
axioms
⋮
error-Attr$_i$ $\ll_{Attr_i}$ undef-Attr$_i$,
copy-Attr$_i$-prd(v-Attr$_i$) → undef-Attr$_i$ $\ll_{Attr_i}$ v-Attr$_i$,
copy-Attr$_i$(v-n-Attr$_i$) $\ll_{Attr_i}$ copy-Attr$_i$(v-n-Attr$_i$-$_1$)
↔ v-n-Attr$_i$ $\ll_{n-Attr_i}$ v-n-Attr$_i$-$_1$,
¬ v-Attr$_i$ $\ll_{Attr_i}$ v-Attr$_i$,
v-Attr$_i$ $\ll_{Attr_i}$ v-Attr$_i$-$_1$ ∧ v-Attr$_i$-$_1$ $\ll_{Attr_i}$ v-Attr$_i$-$_2$
→ v-Attr$_i$ $\ll_{Attr_i}$ v-Attr$_i$-$_2$,
v-Attr$_i$ $\ll_{Attr_i}$ v-Attr$_i$-$_1$
∨ v-Attr$_i$ = v-Attr$_i$-$_1$
∨ v-Attr$_i$-$_1$ $\ll_{Attr_i}$ v-Attr$_i$,
⋮
end enrich

To guarantee the consistency of this enrichment we have to implement the specification *Ordered-Attributes*. This can be done using the generic data specification *Attributes*. A generic data specification is always consistent — assuming the parameter specification is consistent.

4.1.4 The Implementation

Because *OrderedAttributes–Attributes* is an enrichment module we only have to implement the predicates. This implementation is written in a PASCAL-like syntax, and is based on the data types of the specification *Attributes*.

The module:

OrderedAttributes-Attributes =
module
export OrderedAttributes
refinement
representation of operations
⋮
Attr$_i$≪# implements $\ll_{Attr_i}$;
⋮

import Attributes

procedures

⋮

$\text{Attr}_i \ll\#$ $(\text{Attr}_i, \text{Attr}_i)$: bool;

⋮

variables b: bool;

implementation

$\text{Attr}_i \ll\#(\text{v-Attr}_i, \text{v-Attr}_{i\text{-}1};$ **var** b)
begin
if $\text{copy-Attr}_i\text{-prd}(\text{v-Attr}_i) \wedge \text{copy-Attr}_i\text{-prd}(\text{v-Attr}_{i\text{-}1})$ **then**
if $\text{get-Attr}_i(\text{v-Attr}_i) \ll_{n-Attr_i} \text{get-Attr}_i(\text{v-Attr}_{i\text{-}1})$ **then** b := tt **else** b := ff
else
if $(\text{v-Attr}_i = \text{error-Attr}_i \wedge \text{v-Attr}_{i\text{-}1} \neq \text{error-Attr}_i)$
$\vee (\text{v-Attr}_i = \text{undef-Attr}_i \wedge \text{copy-Attr}_i\text{-prd}(\text{v-Attr}_{i\text{-}1}))$ **then**
b := tt
else
b := ff
end

To guarantee the correctness of this implementation and to infer the consistency of the specification *OrderedAttributes* we have to show four classes of proof obligations. In a concrete instantiation e.g. *Cardiac-Catheterisation* these verification conditions were generated automatically by the KIV system.

i- A set of proof obligations which guarantees the termination of the procedures with respect to the used subsets of the import sorts. Termination is necessary because all functions are total. A restriction of the used import sorts is often desirable for example see the implementations of the entity specifications or the implementation of the database specification.

ii- Equality conditions, they are necessary if the equality of the export is not implemented through the equality of the import. These conditions are not necessary in this case study.

iii- Conditions guaranteeing the right behavior of the implemented procedures. For each axiom in the export specification one.

iiii- Conditions guaranteeing hat the used restriction of the import data indeed denotes the set of data reachable by the procedures implementing the export generators.

For a more detailed explanation of the different proof obligations see [Rei92].

4.1.5 Proof Obligations for Attributes

We now present the proof obligations and a sketched proof for guaranteeing the correctness of the implementation. The following DL formulas are formulas in a sequent calculus[1]. They are separated into the different classes described above. Free variables are implicit all quantified.

Conditions for Termination (i)

$\vdash \langle \text{Attr}_i \ll\#(\text{v-Attr}_i, \text{v-Attr}_{i\text{-}1};\text{b})\rangle$ true

Because *OrderedAtttributes-Attributes* is an enrichment module we use the complete input data, no restriction is necessary.

[1] "⊢" is the sequent arrow.

To prove this goal we use symbolic execution. This means we make unfold steps and a case distinction for the if-statement (we "split" the if-statement). Because there is no recursive call in the procedure each branch of the if-statements terminates and so the complete call. This completes the proof.

Proof Obligations Guaranteeing the Right Behavior (iii)

$\vdash \langle \text{Attr}_i {\ll}\#(\text{error-Attr}_i, \text{undef-Attr}_i; \text{b})\rangle\ \text{b} = \text{tt}$

Proof: At first we unfold the right side. Now we can decide the instantiated if conditions. We are working on the else part.

$$\begin{array}{l} \vdash \langle \textbf{if}\ \text{error-Attr}_i = \text{error-Attr}_i \wedge \text{undef-Attr}_i \neq \text{error-Attr}_i \\ \quad\quad \vee\ \text{error-Attr}_i = \text{undef-Attr}_i \wedge \text{copy-Attr}_i\text{-prd}(\text{undef-Attr}_i)\ \textbf{then} \\ \quad\quad \text{b}_0 := \text{tt} \\ \quad \textbf{else} \\ \quad\quad \text{b}_0 := \text{ff}\rangle\, \text{b}_0 = \text{tt} \end{array}$$

The condition is once more decidable and we are working on the then part. An assignment completes the proof.

$\vdash \text{copy-Attr}_i\text{-prd}(\text{v-Attr}_i) \rightarrow \langle \text{Attr}_i {\ll}\#(\text{undef-Attr}_i, \text{v-Attr}_i; \text{b})\rangle\, \text{b} = \text{tt}$

Proof: At first we transform the goal by eliminating the implication.

$$\text{copy-Attr}_i\text{-prd}(\text{v-patids}) \vdash \langle \text{patids} {\ll}\#(\text{undef-Attr}_i, \text{v-Attr}_i; \text{b})\rangle\ \text{b} = \text{tt}$$

Then we unfold the right side. Now we can decide the instantiated if conditions. We are working on the else part. The if condition is decidable too and we switch to the then part. An assignment completes the proof.

$\vdash$
$\langle \text{Attr}_i {\ll}\#(\text{copy-Attr}_i(\text{v-n-Attr}_i), \text{copy-Attr}_i(\text{v-n-Attr}_{i\text{-}1}); \text{b})\rangle\, \text{b} = \text{tt}$
$\leftrightarrow \text{v-n-Attr}_i \ll_{n-Attr_i} \text{v-n-Attr}_{i\text{-}1}$

Proof: At first we transform the goal by eliminating the equivalence. We get two new goals:

1.

$\text{v-n-Attr}_i \ll_{n-Attr_i} \text{v-n-Attr}_{i\text{-}1}$

$\vdash$

$\langle \text{Attr}_i {\ll}\#(\text{copy-Attr}_i(\text{v-n-Attr}_i), \text{copy-Attr}_i(\text{v-n-Attr}_{i\text{-}1}); \text{b})\rangle\ \text{b} = \text{tt}$

2.

$\langle \text{Attr}_i {\ll}\#(\text{copy-Attr}_i(\text{v-n-Attr}_i), \text{copy-Attr}_i(\text{v-n-Attr}_{i\text{-}1}); \text{b})\rangle\ \text{b} = \text{tt},$
$\neg\ \text{v-n-Attr}_i \ll_{n-Attr_i} \text{v-n-Attr}_{i\text{-}1}$

$\vdash$

Both goals can now be closed by symbolic execution and splitting the conditionals.

$\vdash \neg\ \langle \text{Attr}_i {\ll}\#(\text{v-Attr}_i, \text{v-Attr}_i; \text{b})\rangle\, \text{b} = \text{tt}$

Proof: At first we shift the right side of the sequent to the left and then we close the goal by symbolic execution and splitting the conditionals as above.

$\vdash$
$\langle \text{Attr}_i \ll \#(\text{v-Attr}_i,\ \text{v-Attr}_{i\text{-}1};\text{b})\rangle\, \text{b} = \text{tt} \wedge \langle \text{Attr}_i \ll \#(\text{v-Attr}_{i\text{-}1},\ \text{v-Attr}_{i\text{-}2};\text{b})\rangle\, \text{b} = \text{tt}$
$\rightarrow \langle \text{Attr}_i \ll \#(\text{v-Attr}_i,\ \text{v-Attr}_{i\text{-}2};\text{b})\rangle\, \text{b} = \text{tt}$

Proof: At first we eliminate the implication.

$\langle \text{Attr}_i \ll \#(\text{v-Attr}_i,\ \text{v-Attr}_{i\text{-}1};\text{b})\rangle\, \text{b} = \text{tt},\ \langle \text{Attr}_i \ll \#(\text{v-Attr}_{i\text{-}1},\ \text{v-Attr}_{i\text{-}2};\text{b})\rangle\, \text{b} = \text{tt}$
$\vdash$
$\langle \text{Attr}_i \ll \#(\text{v-Attr}_i,\ \text{v-Attr}_{i\text{-}2};\text{b})\rangle\, \text{b} = \text{tt}$

Like above we unfold the right side, this yields two new goals:

1.
$\langle \text{Attr}_i \ll \#(\text{v-Attr}_i,\ \text{v-Attr}_{i\text{-}1};\text{b})\rangle\, \text{b} = \text{tt},\ \langle \text{Attr}_i \ll \#(\text{v-Attr}_{i\text{-}1},\ \text{v-Attr}_{i\text{-}2};\text{b})\rangle\, \text{b} = \text{tt}$
$\neg\ \text{get-Attr}_i(\text{v-Attr}_i) \ll_{n-Attr_i} \text{get-Attr}_i(\text{v-Attr}_{i\text{-}2})$
$\vdash$

2.
$\langle \text{Attr}_i \ll \#(\text{v-Attr}_i,\ \text{v-Attr}_{i\text{-}1};\text{b})\rangle\, \text{b} = \text{tt},\ \langle \text{Attr}_i \ll \#(\text{v-Attr}_{i\text{-}1},\ \text{v-Attr}_{i\text{-}2};\text{b})\rangle\, \text{b} = \text{tt}$
$(\ \text{v-Attr}_{i\text{-}2} = \text{error-Attr}_i$
$\vee\ \text{v-n-Attr}_i \neq \text{error-Attr}_i \wedge \neg\ \text{copy-Attr}_i\text{-prd}(\text{v-Attr}_{i\text{-}2})$
$\vee\ \text{v-n-Attr}_i \neq \text{error-Attr}_i \wedge \neg\ \text{copy-Attr}_i\text{-prd}(\text{v-Attr}_i)$
$\quad \wedge\ \text{v-n-Attr}_i \neq \text{undef-Attr}_i\ )$
$\vdash$

Both goals can now be closed by symbolic execution of the diamond formulas in the antecedent and splitting the conditionals.

$\vdash$
$\langle \text{Attr}_i \ll \#(\text{v-Attr}_i,\ \text{v-Attr}_{i\text{-}1};\text{b})\rangle\, \text{b} = \text{tt}$
$\vee\ \text{v-Attr}_i = \text{v-Attr}_{i\text{-}1}$
$\vee\ \langle \text{Attr}_i \ll \#(\text{v-Attr}_{i\text{-}1},\ \text{v-Attr}_i;\text{b})\rangle\, \text{b} = \text{tt}$

Proof: At first we normalize the goal.

$\text{v-Attr}_i \neq \text{v-Attr}_{i\text{-}1}$
$\vdash$
$\langle \text{Attr}_i \ll \#(\text{v-Attr}_i,\ \text{v-Attr}_{i\text{-}1};\text{b})\rangle\, \text{b} = \text{tt},$
$\langle \text{Attr}_i \ll \#(\text{v-Attr}_{i\text{-}1},\ \text{v-Attr}_i;\text{b})\rangle\, \text{b} = \text{tt}$

Now we unfold the diamond formulas until we get the following two first order formulas which hold in each model of the specification *Attribute*:

1.
$\text{copy-Attr}_i\text{-prd}(\text{v-Attr}_i),\ \text{copy-Attr}_i\text{-prd}(\text{v-Attr}_{i\text{-}1})$
$\vdash$
$\text{v-Attr}_i = \text{v-Attr}_{i\text{-}1},\ \text{get-Attr}_i(\text{v-Attr}_{i\text{-}1}) \ll_{n-Attr_i} \text{get-Attr}_i(\text{v-Attr}_i),$
$\text{get-Attr}_i(\text{v-Attr}_i) \ll_{n-Attr_i} \text{get-Attr}_i(\text{v-Attr}_{i\text{-}1})$

2.
$(\text{v-Attr}_i \neq \text{undef-Attr}_i \wedge \neg\ \text{copy-Attr}_i\text{-prd}(\text{v-Attr}_i)$
$\vee\ \neg\ \text{copy-Attr}_i\text{-prd}(\text{v-Attr}_{i\text{-}1}) \wedge \neg\ \text{copy-Attr}_i\text{-prd}(\text{v-Attr}_i)$
$\vee\ \neg\ \text{copy-Attr}_i\text{-prd}(\text{v-Attr}_{i\text{-}1}) \wedge \text{v-Attr}_{i\text{-}1} \neq \text{undef-Attr}_i\ )$
$\vdash$
$\text{v-Attr}_i = \text{error-Attr}_i,\ \text{v-Attr}_i = \text{v-v-Attr}_{i\text{-}1},\ \text{v-v-Attr}_{i\text{-}1} = \text{error-Attr}_i$

This completes the proof of correctness for the module *OrderedAtttributes-Attributes* and establishes the consistency for the specification *OrderedAtttributes*.

4.2 The Entity Part

The modules *Entity–preEntity(1–n)* are one of the major parts of the translated system. Every module exports a specification *Entity* which provides the sort of one entity type. To ensure the consistency of *Entity* an implementation is generated fully automatically on the basis of the specification *preEntity*, which provides the sort *pre-E* freely generated by the constructors *p-mk-E* and an error constant *p-error-E*. The function *p-mk-E* builds an entity as a tuple of attributes. In addition, a selector function *p-attr* for each attribute is specified. The function *p-mk-E* builds an entity as a tuple of attributes. Furthermore a key sort *p-keysort-E* is specified which combines the mandatory key attributes of the entity.

preEntity$_i$ =
data specification
  using OrderedAttributes
  pre-E$_i$ = p-mk-E$_i$ (p-attr$_{i_1}$: Attr$_{i_1}$,
        p-attr$_{i_n}$: Attr$_{i_n}$)
     | p-error-E$_i$
     ;
  p-keysort-E$_i$ = p-mkkey-E$_i$(k-attr$_{i_k}$:Attr$_{i_k}$, ..., k-attr$_{i_l}$:Attr$_{i_l}$
  variables
    pent$_i$, pent$i$-$_1$: pre-E$_i$;
    pkey$_i$, pkey$i$-$_1$: p-keysort-E$_i$;
end data specification

The sort *E* of the specification *Entity* is like the sort *pre-E* in the specification *preEntity* generated by *error-E* and *create-E*, but not freely. The function *create-E* builds a tuple only if a certain error condition is satisfied. This error condition describes the state if in an entity one or more mandatory attributes are undefined or if at least one attribute is the error attribute. In addition a selector function *attr*, a modify function *set-attr* for each attribute and a lexical order over the key sort *keysort-E* is specified.

Entity$_i$ =
enrich OrderedAttributes **with**
  sorts E$_i$, keysort-E$_i$;
  constants error-E$_i$: E$_i$;
  functions

create-E$_i$	:	Attr$_{i_1}$ × ... × Attr$_{i_n}$	→	E$_i$	;
attr$_{i_1}$	:	E$_i$	→	Attr$_{i_1}$	;
⋮					
attr$_{i_n}$	:	E$_i$	→	Attr$_{i_n}$	;
set-attr$_{i_1}$	:	E$_i$ × Attr$_{i_1}$	→	E$_i$	;
⋮					
set-attr$_{i_n}$	:	E$_i$ × Attr$_{i_n}$	→	E$_i$	;
key-E$_i$	:	E$_i$	→	keysort-E$_i$	;
mkkey-E$_i$	:	Attr$_{i_k}$ × ... × Attr$_{i_l}$	→	keysort-E$_i$	;

  predicates . $\ll_{key-E_i}$. : keysort-E$_i$ × keysort-E$_i$;
  variables
    ent$_i$: E$_i$;
    key$i$-$_2$, key$i$-$_1$, key$_i$: keysort-E$_i$;
    a$_{i_1}$-1, a$_{i_1}$-2: Attr$_{i_1}$;
    ⋮
    a$_{i_n}$-1, a$_{i_n}$-2: Attr$_{i_n}$;
  axioms
    E$_i$ **generated by** create-E$_i$, error-E$_i$;
    keysort-E$_i$ **freely generated by** mkkey-E$_i$;
    a$_{i_j}$-1 = undef-Attr$_{i_1}$ ∨

$\vdots$ *mandatory attributes*
$\vee\ \mathrm{a}_{i_m}\text{-}1 = \text{undef-Attr}_{i_n}$
$\vee\ \mathrm{a}_{i_1}\text{-}1 = \text{error-Attr}_{i_1}\ \vee$
$\vdots$ *all attributes*
$\vee\ \mathrm{a}_{i_n}\text{-}1 = \text{error-Attr}_{i_n}$
$\leftrightarrow$
$\text{create-E}_i(\mathrm{a}_{i_1}\text{-}1, \ldots, \mathrm{a}_{i_n}\text{-}1) = \text{error-E}_i,$
$\text{create-E}_i(\mathrm{a}_{i_1}\text{-}1, \ldots, \mathrm{a}_{i_n}\text{-}1) \neq \text{error-E}_i$
$\rightarrow (\text{create-E}_i(\mathrm{a}_{i_1}\text{-}1, \ldots, \mathrm{a}_{i_n}\text{-}1) = \text{create-E}_i(\mathrm{a}_{i_1}\text{-}2, \ldots, \mathrm{a}_{i_n}\text{-}2)$
$\quad \rightarrow \mathrm{a}_{i_1}\text{-}1 = \mathrm{a}_{i_1}\text{-}2\ \wedge$
$\qquad \vdots$
$\qquad \wedge\ \mathrm{a}_{i_n}\text{-}1 = \mathrm{a}_{i_n}\text{-}2)$
$\quad \wedge\ \text{attr}_{i_1}(\text{create-E}_i(\mathrm{a}_{i_1}\text{-}1, \ldots, \mathrm{a}_{i_n}\text{-}1)) = \mathrm{a}_{i_1}\text{-}1 \wedge$
$\quad \vdots$
$\quad \wedge\ \text{attr}_{i_n}(\text{create-E}_i(\mathrm{a}_{i_1}\text{-}1, \ldots, \mathrm{a}_{i_n}\text{-}1)) = \mathrm{a}_{i_n}\text{-}1,$
$\text{ent}_i \neq \text{error-E}_i$
$\rightarrow \text{set-attr}_{i_1}(\text{ent}_i, \mathrm{a}_{i_1}\text{-}1)$
$\quad = \text{create-E}_i(\mathrm{a}_{i_1}\text{-}1,$
$\qquad\qquad \text{attr}_{i_2}(\text{ent}_i), \ldots,$
$\qquad\qquad \text{attr}_{i_n}(\text{ent}_i))$
$\quad \wedge\ \text{set-attr}_{i_n}(\text{ent}_i, \mathrm{a}_{i_n}\text{-}1)$
$\qquad = \text{create-E}_i(\text{attr}_{i_1}(\text{ent}_i), \ldots,$
$\qquad\quad \text{attr}_{i_{n-1}}(\text{ent}_i),$
$\qquad\quad \mathrm{a}_{i_n}\text{-}1),$
$\text{attr}_{i_1}(\text{error-E}_i) = \text{error-Attr}_{i_1},$
$\vdots$
$\text{attr}_{i_n}(\text{error-E}_i) = \text{error-Attr}_{i_n},$
$\text{set-attr}_{i_1}(\text{error-E}_i, \mathrm{a}_{i_1}\text{-}1) = \text{error-E}_i,$
$\vdots$
$\text{set-attr}_{i_n}(\text{error-E}_i, \mathrm{a}_{i_n}\text{-}1) = \text{error-E}_i,$
$\text{key-E}_i(\text{ent}_i) = \text{mkkey-E}_i(\text{attr}_{i_k}(\text{ent}_i), \ldots, \text{attr}_{i_l}(\text{ent}_i)),$
$\text{mkkey-E}_i(\mathrm{a}_{i_k}\text{-}1, \ldots, \mathrm{a}_{i_l}\text{-}1) \ll_{key-E_i} \text{mkkey-E}_i(\mathrm{a}_{i_k}\text{-}2, \ldots, \mathrm{a}_{i_l}\text{-}2)$
$\leftrightarrow$
$\mathrm{a}_{i_k}\text{-}1 \ll_{Attr_{i_k}} \mathrm{a}_{i_k}\text{-}2\ \vee$
$\vdots$
$\vee\ (\mathrm{a}_{i_k}\text{-}1 = \mathrm{a}_{i_k}\text{-}2 \wedge \ldots \wedge \mathrm{a}_{i_{l-1}}\text{-}1 = \mathrm{a}_{i_{l-1}}\text{-}2 \wedge \mathrm{a}_{i_l}\text{-}1 \ll_{Attr_{i_l}} \mathrm{a}_{i_l}\text{-}2),$
$\neg\ \text{key}_i \ll_{key-E_i} \text{key}_i,$
$\text{key}_i \ll_{key-E_i} \text{key}i\text{-}_{i_1} \wedge \text{key}i\text{-}_{i_1} \ll_{key-E_i} \text{key}i\text{-}_2 \rightarrow \text{key}_i \ll_{key-E_i} \text{key}i\text{-}_2,$
$\text{key}_i \ll_{key-E_i} \text{key}i\text{-}_1 \vee \text{key}_i = \text{key}i\text{-}_1 \vee \text{key}i\text{-}_1 \ll_{key-E_i} \text{key}_i$

end enrich

4.2.1 The Implementation

To guarantee consistency the functions of the specification *Entity* can now be implemented by trivial (flat) programs[2] using the specification *preEntity.* In this implementation the sort E is represented through a subset of *pre-E* containing only the terms *p-error-E* and *p-mk-E(...)* where no mandatory attributes are undefined or no attribute is the error attribute. For the implementation of the sort *keysort-E* the whole sort *p-keysort-E* is used.

[2] By flat we mean programs without recursion and while-loops.

The module:

Entity-preEntity =
module
  export Entity_i
  refinement
    representation of sorts
      pre-E_i implements E_i;
      p-keysort-E_i implements keysort-E_i;
    representation of operations
      error-E_i# implements error-E_i;
      create-E_i# implements create-E_i;
      attr_{i_1}# implements attr_{i_1};
      ⋮
      attr_{i_n}# implements attr_{i_n};
      set-attr_{i_1}# implements set-attr_{i_1};
      ⋮
      set-attr_{i_n}# implements set-attr_{i_n};
      key-E_i# implements key-E_i;
      mkkey-E_i# implements mkkey-E_i;
      key-E_i≪# implements $\ll_{key-E_i}$;

    import preEntity_i

    procedures

error-E_i#	()	: pre-E_i;
create-E_i#	(Attr_{i_1}, ..., Attr_{i_n})	: pre-E_i;
attr_{i_1}#	(pre-E_i)	: Attr_{i_1};
⋮		
attr_{i_n}#	(pre-E_i)	: Attr_{i_n};
set-attr_{i_1}#	(pre-E_i, Attr_{i_1})	: pre-E_i;
⋮		
set-attr_{i_n}#	(pre-E_i, Attr_{i_n})	: pre-E_i;
key-E_i#	(pre-E_i)	: p-keysort-E_i;
mkkey-E_i#	(Attr_{i_k}, ..., Attr_{i_l})	: p-keysort-E_i;
key-E_i≪#	(p-keysort-E_i, p-keysort-E_i)	: bool;

    variables b: bool; a_{i_1}: Attr_{i_1}; ...: ...; a_{i_n}: Attr_{i_n};

    implementation

error-E_i#(**var** pent_i)
begin
  pent_i := p-error-E_i
end

create-E_i#(a_{i_1}, ..., a_{i_n}; **var** pent_i)
begin
  if a_{i_j} = undef-Attr_{i_1}
    $\vee$
    mandatory attributes

    ⋮
    $\vee$ a$_{i_m}$ = undef-Attr$_{i_n}$
    $\vee$ a$_{i_1}$ = error-Attr$_{i_1}$
    $\vee$
    all attributes
    ⋮
    $\vee$ a$_{i_n}$ = error-Attr$_{i_n}$ **then**
   pent$_i$:= p-error-E$_i$
  else
    pent$_i$:= p-mk-E$_i$(a$_{i_1}$, ..., a$_{i_n}$)
end

attr$_{i_1}$#(pent$_i$; **var** a$_{i_1}$)
begin
  if pent$_i$ = p-error-E$_i$ **then** a$_{i_1}$:= error-Attr$_{i_1}$ **else** a$_{i_1}$:= p-attr$_{i_1}$(pent$_i$)
end

  ⋮

attr$_{i_n}$#(pent$_i$; **var** a$_{i_n}$)
begin
  if pent$_i$ = p-error-E$_i$ **then** a$_{i_n}$:= error-Attr$_{i_n}$ **else** a$_{i_n}$:= p-attr$_{i_n}$(pent$_i$)
end

set-attr$_{i_1}$#(pent$_i$, a$_{i_1}$; **var** pent$i$-$_1$)
begin
  if pent$_i$ = p-error-E$_i$ $\vee$ a$_{i_1}$ = error-Attr$_{i_1}$ **then** pent$i$-$_1$:= p-error-E$_i$ **else**
    pent$i$-$_1$:= p-mk-E$_i$(a$_{i_1}$, ..., p-attr$_{i_n}$(pent$_i$))
end

  ⋮

  for mandatory attributes

set-attr$_{i_j}$#(pent$_i$, a$_{i_j}$; **var** pent$i$-$_1$)
begin
  if a$_{i_j}$ = undef-Attr$_{i_j}$ $\vee$ pent$_i$ = p-error-E$_i$ $\vee$ a$_{i_j}$ = error-Attr$_{i_j}$ **then**
    pent$i$-$_1$:= p-error-E$_i$
  else
    pent$i$-$_1$:= p-mk-E$_i$(p-attr$_{i_1}$(pent$_i$), ..., a$_{i_j}$, ..., p-attr$_{i_n}$(pent$_i$))
end

  ⋮

set-attr$_{i_n}$#(pent$_i$, a$_{i_n}$; **var** pent$i$-$_1$)
begin
  if pent$_i$ = p-error-E$_i$ $\vee$ a$_{i_n}$ = error-Attr$_{i_n}$ **then** pent$i$-$_1$:= p-error-E$_i$ **else**
    pent$i$-$_1$:= p-mk-E$_i$(p-attr$_{i_1}$(pent$_i$), ..., a$_{i_n}$)
end

```
key-E_i#(pent_i; var pkey_i)
begin
  if pent_i = p-error-E_i then p-mkkey-E_i(error-Attr_{i_k}, ..., error-Attr_{i_l}) else
   pkey_i := p-mkkey-E_i(p-attr_{i_k}(pent_i), ..., p-attr_{i_l}(pent_i))
end


mkkey-E_i#(a_{i_k}, ..., a_{i_l}; var pkey_i)
begin
  pkey_i := p-mkkey-E_i(a_{i_k}, ..., a_{i_l})
end


key-E_i ≪#(pkey_i, pkeyi-_1; var b)
begin
  if k-attr_{i_k}(pkey_i) ≪_{Attr_{i_k}} k-attr_{i_k}(pkeyi-_1) then b := tt
  else
    ⋮
    if k-attr_{i_k}(pkey_i) = k-attr_{i_k}(pkeyi-_1)
       ∧
       ⋮
       ∧ k-attr_{i_{l-1}}(pkey_i) = k-attr_{i_{l-1}}(pkeyi-_1)
       ∧ k-attr_{i_l}(pkey_i) ≪_{Attr_{i_l}} k-attr_{i_l}(pkeyi-_1) then b := tt
    else
       b := ff
end
```

To guarantee the correctness of this implementation and thereby far the consistency of the specification *Entity* the following verification conditions have to be proved. In a concrete instantiation they were generated automatically.

4.2.2 Proof Obligations

Like above, not all types of proof obligations are necessary. But in contrary to the implementation of *OrderedAttributes* we need in this case a restriction for modeling the used subset of *pre-E*. The aim of this procedure is to terminate if and only if the input data lies in the used subset of *pre-E*, i.e. for each attribute tuple containing error attributes or undefined mandatory attributes the procedure doesn't terminate. For the sort *p-keysort-E* the restriction is the **skip**-statement

restriction

```
rs-E_i#(pent_i)
begin
  if pent_i = p-error-E_i then skip else
   var penti-_1 = p-error-E_i in
   begin
     create-E_i#(p-attr_{i_1}(pent_i),
                  ⋮
                 p-attr_{i_n}(pent_i);
                 penti-_1);
     if penti-_1 = p-error-E_i then abort
   end
end

rs-key-E_i#(pkey_i)
begin skip end
```

Conditions for Termination (i)

For each function and constant of the export specification we formulate a proof obligation which guarantees the termination with respect to the restrictions.

1. Termination of *errorE*$_i$#

   ⊢ ⟨error-E$_i$#(;pent$_i$)⟩ ⟨rs-E$_i$#(pent$_i$)⟩ true

   Proof: At first we unfold the first diamond formula on the right side. This yields:
   ⊢ ⟨rs-E$_i$#(p-error-E$_i$)⟩ true
   by symbolic execution we can close this goal.

2. Termination of *create-E*$_i$#

   ⊢ ⟨create-E$_i$#(a$_{i_1}$-1, ..., a$_{i_n}$-1;pent$_i$)⟩ ⟨rs-E$_i$#(pent$_i$)⟩ true

   Proof: At first we unfold the first diamond formula on the right side. This yields two goals:

   1.
   a$_{i_j}$-1 = undef-Attr$_{i_j}$, ..., a$_{i_m}$-1 = undef-Attr$_{i_m}$,
   a$_{i_1}$-1 = error-Attr$_{i_1}$, ..., a$_{i_n}$-1 = error-Attr$_{i_n}$
   ⊢
   ⟨rs-E$_i$#(p-error-E$_i$)⟩ true

   2.
   a$_{i_j}$-1 ≠ undef-Attr$_{i_j}$, ..., a$_{i_m}$-1 ≠ undef-Attr$_{i_m}$,
   a$_{i_1}$-1 ≠ error-Attr$_{i_1}$, ..., a$_{i_n}$-1 ≠ error-Attr$_{i_n}$
   ⊢
   ⟨rs-E$_i$#(p-mk-E$_i$(a$_{i_1}$-1, ..., a$_{i_n}$-1))⟩ true

   Both goals can be proved through unfolding the right side. All if statements can be decided.

3. Termination of *attr*$_i$#

   ⟨rs-E$_i$#(pent$_i$)⟩ true ⊢ ⟨attr$_{i_i}$#(pent$_i$;a$_{i_i}$-1)⟩ true

   Proof: We need only symbolic executions and splits of the if statements to prove this goal and no unfold step on the left side.

4. Termination of *set-attr*$_i$#. We prove an example for mandatory attributes.

   ⟨rs-E$_i$#(pent$_i$)⟩ true
   ⊢
   ⟨set-attr$_{i_i}$#(pent$_i$, a$_{i_i}$-1;pent*i*-$_1$)⟩ ⟨rs-E$_i$#(pent*i*-$_i$)⟩ true

   Proof: Like above we first unfold the right side and get two new goals:

   1.
   ⟨rs-E$_i$#(pent$_i$)⟩ true
   a$_{i_i}$-1 = error-Attr$_{i_i}$ ∨ a$_{i_i}$-1 = undef-Attr$_{i_i}$ ∨ pent$_i$ = p-error-E$_i$
   ⊢
   ⟨rs-E$_i$#(p-error-E$_i$)⟩ true

   2.
   ⟨rs-E$_i$#(pent$_i$)⟩ true

$a_{i_i}\text{-}1 \neq \text{error-Attr}_{i_i}$, $a_{i_i}\text{-}1 \neq \text{undef-Attr}_{i_i}$, $\text{pent}_i \neq \text{p-error-E}_i$

$\vdash$

$\langle \text{rs-E}_i\#(\text{p-mk-E}_i(\text{p-attr}_{i_1}(\text{pent}_i), \ldots, a_{i_i}\text{-}1, \ldots, \text{p-attr}_{i_n}(\text{pent}_i))) \rangle$ true

The proof of the first goal is clear. For the second goal we unfold the left side until we get the following goal:

$a_{i_i}\text{-}1 \neq \text{error-Attr}_{i_i}$, $a_{i_i}\text{-}1 \neq \text{undef-Attr}_{i_i}$, $\text{pent}_i \neq \text{p-error-E}_i$,
$\text{p-attr}_{i_1}(\text{pent}_i) \neq \text{error-Attr}_{i_1}$, …, $\text{p-attr}_{i_n}(\text{pent}_i) \neq \text{error-Attr}_{i_n}$,
$\text{p-attr}_{i_j}(\text{pent}_i) \neq \text{undef-Attr}_{i_j}$, …, $\text{p-attr}_{i_m}(\text{pent}_i) \neq \text{undef-Attr}_{i_m}$,

$\vdash$

$\langle \text{rs-E}_i\#(\text{p-mk-E}_i(\text{p-attr}_{i_1}(\text{pent}_i), \ldots, a_{i_i}\text{-}1, \ldots, \text{p-attr}_{i_n}(\text{pent}_i))) \rangle$ true

Now we unfold the right side and close this goal. All if statements can be decided.

5. Termination of *key-E$_i$#*

$\langle \text{rs-E}_i\#(\text{pent}_i) \rangle$ true

$\vdash$

$\langle \text{key-E}_i\#(\text{pent}_i;\text{pkey}_i) \rangle\ \langle \text{rs-key-E}_i\#(\text{pkey}_i) \rangle$ true

Proof: Even like above a trivial symbolic execution and one split of the if statement followed by unfold steps and we close this goal.

6. Termination of *mkkey-E$_i$#*

$\vdash \langle \text{mkkey-E}_i\#(a_{i_k}\text{-}1, \ldots, a_{i_l}\text{-}1;\text{pkey}_i) \rangle\ \langle \text{rs-key-E}_i\#(\text{pkey}_i) \rangle$ true

Proof: In a more trivial way we close this goal by repeated calls and assignments.

7. Termination of *key-E$_i$≪#*

$\langle \text{rs-key-E}_i\#(\text{pkey}_i) \rangle$ true, $\langle \text{rs-key-E}_i\#(\text{pkey}i\text{-}_1) \rangle$ true

$\vdash$

$\langle \text{key-E}_i{\ll}\#(\text{pkey}_i, \text{pkey}i\text{-}_1;\text{b}) \rangle$ true

Proof: We close this goal by one unfold step on the right side. A split of the if statement and two assignments.

Proof Obligations Guaranteeing the Right Behavior (iii)

1. Defindedness of *create-E$_i$*

$$
\begin{aligned}
\vdash\ & \\
& a_{i_j}\text{-}1 = \text{undef-Attr}_{i_j} \\
& \lor \\
& \vdots \\
& \lor\ a_{i_m}\text{-}1 = \text{undef-Attr}_{i_m} \\
& \lor\ a_{i_1}\text{-}1 = \text{error-Attr}_{i_1} \\
& \lor \\
& \vdots \\
& \lor\ a_{i_n}\text{-}1 = \text{error-Attr}_{i_n} \\
\leftrightarrow\ & \langle \text{create-E}_i\#(a_{i_1}\text{-}1, \ldots, a_{i_n}\text{-}1;\text{pent}_{i0}) \rangle \\
& \langle \text{error-E}_i\#(;\text{pent}_{i1}) \rangle\, \text{pent}_{i0} = \text{pent}_{i1}
\end{aligned}
$$

Proof: At first we normalize the goal by eliminating the equivalence and we get two new goals:

1.

$\langle$create-E_i#(a_{i_1}-1, ..., a_{i_n}-1;$pent_{i\,0}$)$\rangle$
$\langle$error-E_i#(;$pent_{i\,1}$)$\rangle$ $pent_{i\,0}$ = $pent_{i\,1}$
a_{i_j}-1 $\neq$ undef-$Attr_{i_j}$, ..., a_{i_m}-1 $\neq$ undef-$Attr_{i_m}$,
a_{i_1}-1 $\neq$ error-$Attr_{i_1}$, ..., a_{i_n}-1 $\neq$ error-$Attr_{i_n}$
$\vdash$

2.

(a_{i_j}-1 = undef-$Attr_{i_j}$ $\vee$ ... $\vee$ a_{i_m}-1 = undef-$Attr_{i_m}$
$\vee$ a_{i_1}-1 = error-$Attr_{i_1}$ $\vee$ ... $\vee$ a_{i_n}-1 = error-$Attr_{i_n}$
$\vdash$
$\langle$create-E_i#(a_{i_1}-1, ..., a_{i_n}-1;$pent_{i\,0}$)$\rangle$
$\langle$error-E_i#(;$pent_{i\,1}$)$\rangle$ $pent_{i\,0}$ = $pent_{i\,1}$

To prove the first goal we call the first diamond formula to instantiate $pent_{i\,0}$. The result for this instantiation is:

$$pent_{i\,0} = \text{p-mk-}E_i(a_{i_1}\text{-1}, \ldots, a_{i_n}\text{-1})$$

Now we call the second diamond an close the first goal.

To prove the second goal we call the first left side diamond. All the if statements can be decided and we get *p-error-E_i* as instantiation for $pent_{i\,0}$. With this instantiation we call the second diamond and finish the proof.

2. Uniqueness of entities and selecting attributes.

$\vdash$
  $\neg$ $\langle$create-E_i#(a_{i_1}-1, ..., a_{i_n}-1;$pent_{i\,0}$)$\rangle$
    $\langle$error-E_i#(;$pent_{i\,1}$)$\rangle$ $pent_{i\,0}$ = $pent_{i\,1}$
$\rightarrow$ ($\langle$create-E_i#(a_{i_1}-1, ..., a_{i_n}-1;$pent_{i\,0}$)$\rangle$
    $\langle$create-E_i#(a_{i_1}-2, ..., a_{i_n}-2;$pent_{i\,1}$)$\rangle$ $pent_{i\,0}$ = $pent_{i\,1}$
    $\rightarrow$ a_{i_1}-1 = a_{i_1}-2 $\wedge$ ... $\wedge$ a_{i_n}-1 = a_{i_n}-2)
  $\wedge$ $\langle$create-E_i#(a_{i_1}-1, ..., a_{i_n}-1;$pent_{i\,0}$)$\rangle$
    $\langle$$attr_{i_1}$#($pent_{i\,0}$;$a_{i_1}$-$1_0$)$\rangle$ a_{i_1}-1_0 = a_{i_1}-1
  $\wedge$
  $\vdots$
  $\wedge$ $\langle$create-E_i#(a_{i_1}-1, ..., a_{i_n}-1;$pent_{i\,0}$)$\rangle$
    $\langle$$attr_{i_n}$#($pent_{i\,0}$;$a_{i_n}$-$1_0$)$\rangle$ a_{i_n}-1_0 = a_{i_n}-1

Proof: At first we normalize the goal. This yields two classes of subgoals:

1.

$\langle$create-E_i#(a_{i_1}-1, ..., a_{i_n}-1;$pent_{i\,0}$)$\rangle$ $\langle$create-E_i#(a_{i_1}-2, ..., a_{i_n}-2;$pent_{i\,1}$)$\rangle$ $pent_{i\,0}$ = $pent_{i\,1}$,
$\neg$ (a_{i_1}-1 = a_{i_1}-2 $\wedge$ ... $\wedge$ a_{i_n}-1 = a_{i_n}-2)
$\vdash$
$\langle$create-E_i#(a_{i_1}-1, ..., a_{i_n}-1;$pent_{i\,0}$)$\rangle$ $\langle$error-E_i#(;$pent_{i\,1}$)$\rangle$ $pent_{i\,0}$ = $pent_{i\,1}$

2. For each selector function $attr_{i_i}$ one of the following form:

$\vdash$
$\langle$create-E_i#(a_{i_1}-1, ..., a_{i_n}-1;$pent_{i\,0}$)$\rangle$ $\langle$$attr_{i_i}$#($pent_{i\,0}$;$a_{i_i}$-$1_0$)$\rangle$ a_{i_i}-1_0 = a_{i_i}-1
$\langle$create-E_i#(a_{i_1}-1, ..., a_{i_n}-1;$pent_{i\,0}$)$\rangle$ $\langle$error-E_i#(;$pent_{i\,1}$)$\rangle$ $pent_{i\,0}$ = $pent_{i\,1}$

Working on the first goal. We eliminate the first diamond on the left side by introducing a new variable which represent the value of its call.

$\langle \text{create-E}_i\#(\text{a}_{i_1}\text{-}_2, \ldots, \text{a}_{i_n}\text{-}_2;\text{pent}_{i\,1})\rangle\ \text{pent}_{i\,2} = \text{pent}_{i\,1},$
$\langle \text{create-E}_i\#(\text{a}_{i_1}\text{-}_1, \ldots, \text{a}_{i_n}\text{-}_1;\text{pent}_{i\,0})\rangle\ \text{pent}_{i\,0} = \text{pent}_{i\,2},$
$\neg\ (\text{a}_{i_1}\text{-}_1 = \text{a}_{i_1}\text{-}_2 \wedge \ldots \wedge \text{a}_{i_n}\text{-}_1 = \text{a}_{i_n}\text{-}_2)$
$\vdash$
$\langle \text{error-E}_i\#(;\text{pent}_{i\,1})\rangle\, \text{pent}_{i\,2} = \text{pent}_{i\,1}$

Then we call the right side this yields:

$\langle \text{create-E}_i\#(\text{a}_{i_1}\text{-}_2, \ldots, \text{a}_{i_n}\text{-}_2;\text{pent}_{i\,1})\rangle\ \text{pent}_{i\,2} = \text{pent}_{i\,1},$
$\langle \text{create-E}_i\#(\text{a}_{i_1}\text{-}_1, \ldots, \text{a}_{i_n}\text{-}_1;\text{pent}_{i\,0})\rangle\ \text{pent}_{i\,0} = \text{pent}_{i\,2},$
$\neg\ (\text{a}_{i_1}\text{-}_1 = \text{a}_{i_1}\text{-}_2 \wedge \ldots \wedge \text{a}_{i_n}\text{-}_1 = \text{a}_{i_n}\text{-}_2),$
$\text{pent}_{i\,2} \neq \text{p-error-E}_i$
$\vdash$

Now we close this goal by repeated calls, if splits and assignments on the left side. For each goal of the second class we do the following. At first we eliminate the call of create. We get the following goal:

$[\text{create-E}_i\#(\text{a}_{i_1}\text{-}_1, \ldots, \text{a}_{i_n}\text{-}_1;\text{pent}_{i\,0})]\ \text{pent}_{i\,0} = \text{pent}_{i\,2}$
$\vdash$
$\langle \text{error-E}_i\#(;\text{pent}_{i\,1})\rangle\, \text{pent}_{i\,2} = \text{pent}_{i\,1}$
$\langle \text{attr}_{i_i}\#(\text{pent}_{i\,2};\text{a}_{i_i}\text{-}_{1\,0})\rangle\, \text{a}_{i_i}\text{-}_{1\,0} = \text{a}_{i_i}\text{-}_1$

Then we call the right side diamonds this yields:
$[\text{create-E}_i\#(\text{a}_{i_1}\text{-}_1, \ldots, \text{a}_{i_n}\text{-}_1;\text{pent}_{i\,0})]\ \text{pent}_{i\,0} = \text{pent}_{i\,2},$
$\text{pent}_{i\,2} \neq \text{p-error-E}_i,\ \text{p-attr}_{i_i}(\text{pent}_{i\,2}) \neq \text{a}_{i_i}\text{-}_1$
$\vdash$

Finally we call the left side box split the if and close both new goals after one assignment.

3. The correspondence between $\text{set-attr}_{i_i}\#$ and $\text{attr}_{i_i}\#$

$\langle \text{rs-E}_i\#(\text{pent}_i)\rangle\ \text{true}$
$\vdash$
$\quad \neg\ \langle \text{error-E}_i\#(;\text{pent}_{i\,0})\rangle\, \text{pent}_i = \text{pent}_{i\,0}$
$\rightarrow \langle \text{set-attr}_{i_1}\#(\text{pent}_i,\ \text{a}_{i_1}\text{-}_1;\text{pent}i\text{-}_1)\rangle$
$\quad \langle \text{attr}_{i_2}\#(\text{pent}_i;\text{a}_{i_2}\text{-}_{1\,0})\rangle$
$\quad \vdots$
$\quad \langle \text{attr}_{i_n}\#(\text{pent}_i;\text{a}_{i_n}\text{-}_{1\,0})\rangle$
$\quad \langle \text{create-E}_i\#(\text{a}_{i_1}\text{-}_1, \ldots, \text{a}_{i_n}\text{-}_{1\,0};\text{pent}_{i\,0})\rangle\, \text{pent}i\text{-}_1 = \text{pent}_{i\,0}$
$\quad \wedge$
$\quad \vdots$
$\quad \wedge\ \langle \text{set-attr}_{i_n}\#(\text{pent}_i,\ \text{a}_{i_n}\text{-}_1;\text{pent}i\text{-}_1)\rangle$
$\quad\quad \langle \text{attr}_{i_1}\#(\text{pent}_i;\text{a}_{i_1}\text{-}_{1\,0})\rangle$
$\quad\quad \langle \text{attr}_{i_{n-1}}\#(\text{pent}_i;\text{a}_{i_{n-1}}\text{-}_{1\,0})\rangle$
$\quad\quad \langle \text{create-E}_i\#(\text{a}_{i_1}\text{-}_{1\,0}, \ldots, \text{a}_{i_n}\text{-}_1;\text{pent}_{i\,0})\rangle\, \text{pent}i\text{-}_1 = \text{pent}_{i\,0}$

Proof: At first we normalize this goal and get n sub goals, for each *set-attr* function one of the following form:

$\langle \text{rs-E}_i\#(\text{pent}_i)\rangle\ \text{true}$
$\vdash$
$\langle \text{error-E}_i\#(;\text{pent}_{i\,0})\rangle\, \text{pent}_i = \text{pent}_{i\,0},$
$\langle \text{set-attr}_{i_i}\#(\text{pent}_i,\ \text{a}_{i_i}\text{-}_1;\text{pent}i\text{-}_1)\rangle$
$\langle \text{attr}_{i_1}\#(\text{pent}_i;\text{a}_{i_i}\text{-}_{1\,0})\rangle$

$\vdots$

$\langle \text{attr}_{i_{i-1}} \#(\text{pent}_i;\text{a}_{i_i}\text{-}1_0)\rangle$
$\langle \text{attr}_{i_{i+1}} \#(\text{pent}_i;\text{a}_{i_i}\text{-}1_0)\rangle$

$\vdots$

$\langle \text{attr}_{i_n} \#(\text{pent}_i;\text{a}_{i_n}\text{-}1_0)\rangle$
$\langle \text{create-E}_i \#(\text{a}_{i_n}\text{-}1_0, \ldots, \text{a}_{i_i}\text{-}1, \ldots, \text{a}_{i_n}\text{-}1_0;\text{pent}_{i\,0})\rangle\, \text{pent}i\text{-}_1 = \text{pent}_{i\,0}$

To close such goals we unfold the first diamond on the right side and get the following goal:

$\langle \text{rs-E}_i \#(\text{pent}_i)\rangle$ true, $\text{pent}_i \neq \text{p-error-E}_i$
$\vdash$
$\langle \text{set-attr}_{i_i} \#(\text{pent}_i, \text{a}_{i_i}\text{-}1;\text{pent}i\text{-}_1)\rangle$
$\langle \text{attr}_{i_1} \#(\text{pent}_i;\text{a}_{i_i}\text{-}1_0)\rangle$

$\vdots$

$\langle \text{attr}_{i_{i-1}} \#(\text{pent}_i;\text{a}_{i_i}\text{-}1_0)\rangle$
$\langle \text{attr}_{i_{i+1}} \#(\text{pent}_i;\text{a}_{i_i}\text{-}1_0)\rangle$

$\vdots$

$\langle \text{attr}_{i_n} \#(\text{pent}_i;\text{a}_{i_n}\text{-}1_0)\rangle$
$\langle \text{create-E}_i \#(\text{a}_{i_n}\text{-}1_0, \ldots, \text{a}_{i_i}\text{-}1, \ldots, \text{a}_{i_n}\text{-}1_0;\text{pent}_{i\,0})\rangle\, \text{pent}i\text{-}_1 = \text{pent}_{i\,0}$

We now unfold the first diamond on the left side once more. This yields two new goals were the term $\text{a}_{i_i}\text{-}1 = \text{undef-Attr}_{i_i}$ is omitted if the attribute is not a mandatory attribute.

1.
$\langle \text{rs-E}_i \#(\text{pent}_i)\rangle$ true, $\text{pent}_i \neq \text{p-error-E}_i$,
$\text{a}_{i_i}\text{-}1 = \text{undef-Attr}_{i_i} \vee \text{a}_{i_i}\text{-}1 = \text{error-Attr}_{i_i} \vee \text{pent}_i = \text{p-error-E}_i$
$\vdash$
$\langle \text{attr}_{i_1} \#(\text{pent}_i;\text{a}_{i_i}\text{-}1_0)\rangle$

$\vdots$

$\langle \text{attr}_{i_{i-1}} \#(\text{pent}_i;\text{a}_{i_i}\text{-}1_0)\rangle$
$\langle \text{attr}_{i_{i+1}} \#(\text{pent}_i;\text{a}_{i_i}\text{-}1_0)\rangle$

$\vdots$

$\langle \text{attr}_{i_n} \#(\text{pent}_i;\text{a}_{i_n}\text{-}1_0)\rangle$
$\langle \text{create-E}_i \#(\text{a}_{i_n}\text{-}1_0, \ldots, \text{a}_{i_i}\text{-}1, \ldots, \text{a}_{i_n}\text{-}1_0;\text{pent}_{i\,0})\rangle\, \text{p-error-E}_i = \text{pent}_{i\,0}$

2.
$\langle \text{rs-E}_i \#(\text{pent}_i)\rangle$ true, $\text{pent}_i \neq \text{p-error-E}_i$,
$\text{a}_{i_i}\text{-}1 \neg \text{undef-Attr}_{i_i}, \text{a}_{i_i}\text{-}1 \neg \text{error-Attr}_{i_i}, \text{pent}_i \neg \text{p-error-E}_i$
$\vdash$
$\langle \text{attr}_{i_1} \#(\text{pent}_i;\text{a}_{i_i}\text{-}1_0)\rangle$

$\vdots$

$\langle \text{attr}_{i_{i-1}} \#(\text{pent}_i;\text{a}_{i_i}\text{-}1_0)\rangle$
$\langle \text{attr}_{i_{i+1}} \#(\text{pent}_i;\text{a}_{i_i}\text{-}1_0)\rangle$

$\vdots$

$\langle \text{attr}_{i_n} \#(\text{pent}_i;\text{a}_{i_n}\text{-}1_0)\rangle$
$\langle \text{create-E}_i \#(\text{a}_{i_n}\text{-}1_0, \ldots, \text{a}_{i_i}\text{-}1, \ldots, \text{a}_{i_n}\text{-}1_0;\text{pent}_{i\,0})\rangle$

p-mk-E_i(p-$attr_{i_1}$($pent_i$), ..., a_{i_i}-1, ..., (p-$attr_{i_n}$) = $pent_{i0}$

In both cases we call the right side until we get for the first goal:

$\langle$rs-E_i#($pent_i$)$\rangle$ true, $pent_i \neq$ p-error-E_i,
a_{i_i}-1 = undef-$Attr_{i_i}$ $\vee$ a_{i_i}-1 = error-$Attr_{i_i}$ $\vee$ $pent_i$ = p-error-E_i
$\vdash$
$\langle$create-E_i#(p-$attr_{i_1}$($pent_i$), ...,
a_{i_i}-1, ...,
p-$attr_{i_n}$($pent_i$);$pent_{i0}$)$\rangle$
p-error-E_i = $pent_{i0}$

and for the second goal:

$\langle$rs-E_i#($pent_i$)$\rangle$ true, $pent_i \neq$ p-error-E_i,
a_{i_i}-1 $\neq$ undef-$Attr_{i_i}$, a_{i_i}-1 $\neq$ error-$Attr_{i_i}$, $pent_i \neq$ p-error-E_i
$\vdash$
$\langle$create-E_i#(p-$attr_{i_1}$($pent_i$), ...,
a_{i_i}-1, ...,
p-$attr_{i_n}$($pent_i$);$pent_{i0}$)$\rangle$
p-mk-E_i(p-$attr_{i_1}$($pent_i$), ...,
a_{i_i}-1, ...,
p-$attr_{i_n}$($pent_i$)) = $pent_{i0}$

Now we call in both cases the diamond on the right side and close these goals.

4. The following two goals deals with error propagation. For proving these goals we call the diamonds in order of their appearance, the two if tests which are necessary can be decided In both cases we are working on the then part. after the assignment we close the first verification condition. For closing the second we call once more the procedure *error-E_i#*.

   1.

   $\vdash$ $\langle$error-E_i#(;$pent_{i0}$)$\rangle$ $\langle$$attr_{i_1}$#($pent_{i0}$;$a_{i_1}$-$1_0$)$\rangle$ a_{i_1}-1_0 = error-$Attr_{i_1}$

   2.

   $\vdash$
   $\langle$error-E_i#(;$pent_{i0}$)$\rangle$ $\langle$set-$attr_{i_1}$#($pent_{i0}$, a_{i_1}-1;pent$i$-$_1$)$\rangle$
   $\langle$error-E_i#(;$pent_{i1}$)$\rangle$ pent$i$-$_1$ = $pent_{i1}$

5. The correspondents between key-E_i# and mkkey-E_i#

   $\langle$rs-E_i#($pent_i$)$\rangle$ true
   $\vdash$
   $\langle$key-E_i#($pent_i$;$pkey_{i0}$)$\rangle$ $\langle$$attr_{i_k}$#($pent_i$;$a_{i_k}$-$1_0$)$\rangle$ ... $\langle$$attr_{i_l}$#($pent_i$;$a_{i_l}$-$1_0$)$\rangle$
   $\langle$mkkey-E_i#(a_{i_k}-1_0, ..., a_{i_l}-1_0;$pkey_{i1}$)$\rangle$ $pkey_{i0}$ = $pkey_{i1}$

   Proof: At first we call the diamond $\langle$key-E_i#($pent_i$;$pkey_{i0}$)$\rangle$ and then the procedures selecting the attributes. This yields two new goals:

   1.
   $\langle$rs-E_i#($pent_i$)$\rangle$ true,
   $pent_i$ = p-error-E_i

$\vdash$
$\langle$mkkey-E_i#(error-$Attr_{i_k}$, ..., error-$Attr_{i_l}$; $pkey_{i\,1}$)$\rangle$
p-mkkey-E_i(error-$Attr_{i_k}$, ..., error-$Attr_{i_l}$) = $pkey_{i\,1}$

2.
$\langle$rs-E_i#($pent_i$)$\rangle$ true,
$pent_i \neq$ p-error-E_i
$\vdash$
$\langle$key-E_i#($pent_i$;$pkey_{i\,0}$)$\rangle$ $\langle attr_{i_k}$#($pent_i$;a_{i_k}-1_0)$\rangle$ ... $\langle attr_{i_l}$#($pent_i$;a_{i_l}-1_0)$\rangle$
$\langle$mkkey-E_i#(p-$attr_{i_k}$($pent_i$), ..., p-$attr_{i_l}$($pent_i$);$pkey_{i\,1}$)$\rangle$
p-mkkey-E_i((p-$attr_{i_k}$($pent_i$), ..., p-$attr_{i_l}$($pent_i$)) = $pkey_{i\,1}$

In both cases we unfold the right side once more and close the goal.

6. The right implementation of mkkey-E_i#.

$\vdash$
$\langle$mkkey-E_i#(a_{i_k}-1, ..., a_{i_l}-1;$pkey_{i\,0}$)$\rangle$ $\langle$mkkey-E_i#(a_{i_k}-2, ..., a_{i_l}-2;$pkey_{i\,1}$)$\rangle$
$\langle$key-$E_i$$\ll$#($pkey_{i\,0}$, $pkey_{i\,1}$;b)$\rangle$ b = tt
$\leftrightarrow a_{i_k}\text{-}1 \ll_{Attr_{i_k}} a_{i_k}\text{-}2 \lor$
$\vdots$
$\lor (a_{i_k}\text{-}1 = a_{i_k}\text{-}2 \land \ldots \land a_{i_{l-1}}\text{-}1 = a_{i_{l-1}}\text{-}2 \land a_{i_l}\text{-}1 \ll_{Attr_{i_l}} a_{i_l}\text{-}2)$

Proof: At first we eliminate the equivalence. This yields the following two goals:

1.
$a_{i_k}\text{-}1 \ll_{Attr_{i_k}} a_{i_k}\text{-}2 \lor$
$\vdots$
$\lor (a_{i_k}\text{-}1 = a_{i_k}\text{-}2 \land \ldots \land a_{i_{l-1}}\text{-}1 = a_{i_{l-1}}\text{-}2 \land a_{i_l}\text{-}1 \ll_{Attr_{i_l}} a_{i_l}\text{-}2)$
$\vdash$
$\langle$mkkey-E_i#(a_{i_k}-1, ..., a_{i_l}-1;$pkey_{i\,0}$)$\rangle$ $\langle$mkkey-E_i#(a_{i_k}-2, ..., a_{i_l}-2;$pkey_{i\,1}$)$\rangle$
$\langle$key-$E_i$$\ll$#($pkey_{i\,0}$, $pkey_{i\,1}$;b)$\rangle$ b = tt

2.
$\langle$mkkey-E_i#(a_{i_k}-1, ..., a_{i_l}-1;$pkey_{i\,0}$)$\rangle$ $\langle$mkkey-E_i#(a_{i_k}-2, ..., a_{i_l}-2;$pkey_{i\,1}$)$\rangle$
$\langle$key-$E_i$$\ll$#($pkey_{i\,0}$, $pkey_{i\,1}$;b)$\rangle$ b = tt,
$\neg\, a_{i_k}\text{-}1 \ll_{Attr_{i_k}} a_{i_k}\text{-}2,$
$\vdots$
$\neg(a_{i_k}\text{-}1 = a_{i_k}\text{-}2 \land \ldots \land a_{i_{l-1}}\text{-}1 = a_{i_{l-1}}\text{-}2 \land a_{i_l}\text{-}1 \ll_{Attr_{i_l}} a_{i_l}\text{-}2)$
$\vdash$

In both cases we call the procedures *mkkey-E_i#* and reduce the goals to:

1.
$a_{i_k}\text{-}1 \ll_{Attr_{i_k}} a_{i_k}\text{-}2 \lor$
$\vdots$
$\lor (a_{i_k}\text{-}1 = a_{i_k}\text{-}2 \land \ldots \land a_{i_{l-1}}\text{-}1 = a_{i_{l-1}}\text{-}2 \land a_{i_l}\text{-}1 \ll_{Attr_{i_l}} a_{i_l}\text{-}2)$
$\vdash$
$\langle$key-$E_i$$\ll$#(p-mk-key-$E_i$($a_{i_k}$-1, ..., a_{i_l}-1), p-mk-key-E_i(a_{i_k}-2, ..., a_{i_l}-2);b)$\rangle$ b = tt

2.

$\langle$key-$E_i \ll \#$(p-mk-key-E_i(a_{i_k}-1, ..., a_{i_l}-1), p-mk-key-E_i(a_{i_k}-2, ..., a_{i_l}-2);b)$\rangle$ b = tt,

$\neg$ a_{i_k}-1 $\ll_{Attr_{i_k}}$ a_{i_k}-2,

$\vdots$

$\neg$(a_{i_k}-1 = a_{i_k}-2 $\wedge \ldots \wedge$ $a_{i_{l-1}}$-1 = $a_{i_{l-1}}$-2 $\wedge$ a_{i_l}-1 $\ll_{Attr_{i_l}}$ a_{i_l}-2)

$\vdash$

Now we call the last diamond and close both goals.

7. The following three proof obligations guaranteeing the total order over the keys.

1.

$\langle$rs-key-$E_i \#$(pkey$_i$)$\rangle$ true $\vdash$ $\neg$ $\langle$key-$E_i \ll \#$(pkey$_i$, pkey$_i$;b)$\rangle$ b = tt

Proof: We normalize this goal by shifting the diamond on the right side to the left and omitting the negation. Then we unfold this diamond formula. The if can be decided and we are working on the then part and close this goal by an assignment.

2.

$\langle$rs-key-$E_i \#$(pkey$i$-$_0$)$\rangle$ true,

$\langle$rs-key-$E_i \#$(pkey$_i$)$\rangle$ true,

$\langle$rs-key-$E_i \#$(pkey$i$-$_1$)$\rangle$ true

$\vdash$

$\langle$key-$E_i \ll \#$(pkey$_i$, pkey$i$-$_0$;b)$\rangle$ b = tt $\wedge$ $\langle$key-$E_i \ll \#$(pkey$i$-$_0$, pkey$i$-$_1$;b)$\rangle$ b = tt
$\rightarrow$ $\langle$key-$E_i \ll \#$(pkey$_i$, pkey$i$-$_1$;b)$\rangle$ b = tt

Proof: At first we eliminate the implication.

$\langle$key-$E_i \ll \#$(pkey$_i$, pkey$i$-$_0$;b)$\rangle$ b = tt,
$\langle$key-$E_i \ll \#$(pkey$i$-$_0$, pkey$i$-$_1$;b)$\rangle$ b = tt
$\langle$rs-key-$E_i \#$(pkey$i$-$_0$)$\rangle$ true,
$\langle$rs-key-$E_i \#$(pkey$_i$)$\rangle$ true,
$\langle$rs-key-$E_i \#$(pkey$i$-$_1$)$\rangle$ true,
$\vdash$
$\langle$key-$E_i \ll \#$(pkey$_i$, pkey$i$-$_1$;b)$\rangle$ b = tt

Like above we unfold the right side, then we split the if cascade. This yields new goals which can be closed by an assignment. Only one goal is still open.

$\langle$key-$E_i \ll \#$(pkey$_i$, pkey$i$-$_0$;b)$\rangle$ b = tt,
$\langle$key-$E_i \ll \#$(pkey$i$-$_0$, pkey$i$-$_1$;b)$\rangle$ b = tt
$\langle$rs-key-$E_i \#$(pkey$i$-$_0$)$\rangle$ true,
$\langle$rs-key-$E_i \#$(pkey$_i$)$\rangle$ true,
$\langle$rs-key-$E_i \#$(pkey$i$-$_1$)$\rangle$ true,
$\neg$ k-attr$_{i_k}$(pkey$_i$) $\ll_{Attr_{i_k}}$ k-attr$_{i_k}$(pkey$i$-$_1$), ...,
$\neg$ (k-attr$_{i_k}$(pkey$_i$) = k-attr$_{i_k}$(pkey$i$-$_1$) $\wedge \ldots$
$\wedge$ k-attr$_{i_{l-1}}$(pkey$_i$) = k-attr$_{i_{l-1}}$(pkey$i$-$_1$)
$\wedge$ k-attr$_{i_l}$(pkey$_i$) $\ll_{Attr_{i_l}}$ k-attr$_{i_l}$(pkey$i$-$_1$))
$\vdash$

Now we unfold the first two diamond formulas on the left side and close the goal.

3.

$\langle$rs-key-E$_i$#(pkey$_i$)$\rangle$ true,
$\langle$rs-key-E$_i$#(pkey$i$-$_1$)$\rangle$ true
$\vdash$
$\langle$key-E$_i$$\ll$#(pkey$i$-$_1$, pkey$_i$;b)$\rangle$ b = tt
$\vee$ pkey$i$-$_1$ = pkey$_i$
$\vee$ $\langle$key-E$_i$$\ll$#(pkey$_i$, pkey$i$-$_1$;b)$\rangle$ b = tt

Proof: At first we normalize the goal and call the first diamond on the left side. This yields the following new goal:

$\langle$rs-key-E$_i$#(pkey$_i$)$\rangle$ true,
$\langle$rs-key-E$_i$#(pkey$i$-$_1$)$\rangle$ true,
pkey$i$-$_1$ $\neg$ pkey$_i$,
$\neg$ k-attr$_{i_k}$(pkey$_i$) $\ll_{Attr_{i_k}}$ k-attr$_{i_k}$(pkey$i$-$_1$), ...,
$\neg$ (k-attr$_{i_k}$(pkey$_i$) = k-attr$_{i_k}$(pkey$i$-$_1$) $\wedge$ ...
$\wedge$ k-attr$_{i_{l-1}}$(pkey$_i$) = k-attr$_{i_{l-1}}$(pkey$i$-$_1$)
$\wedge$ k-attr$_{i_l}$(pkey$_i$) $\ll_{Attr_{i_l}}$ k-attr$_{i_l}$(pkey$i$-$_1$))
$\vdash$
$\langle$key-E$_i$$\ll$#(pkey$_i$, pkey$i$-$_1$;b)$\rangle$ b = tt

We unfold the left side again. This yields the same first order formula on the right side as the first unfold step but with rotated variables. Combined with pkey$i$-$_1$ $\neq$ pkey$_i$ this is a contradiction and we can finish the proof.

8. This proof obligation guarantees the freely axiom for mkkey-E$_i$.

$\vdash$
$\langle$mkkey-E$_i$#(v-Attr$_{i_k}$, ..., v-Attr$_{i_l}$;pkey$_{i\,0}$)$\rangle$ $\langle$mkkey-E$_i$#(v-Attr$_{i_{1\,0}}$, ..., v-Attr$_{i_{l\,0}}$;pkey$_{i\,1}$)$\rangle$
pkey$_{i\,0}$ = pkey$_{i\,1}$
$\leftrightarrow$ v-Attr$_{i_k}$ = v-Attr$_{i_{k\,0}}$ $\wedge$ ... $\wedge$ v-Attr$_{i_l}$ = v-Attr$_{i_{l\,0}}$

Proof: We normalize this goal and get two new goals then we unfold the procedures and use the freely axioms for *p-mkkey-E$_i$* and we finish the proof.

Conditions for Restrictions (iiii)

Before we are going to prove, we present the restrictions which are generated in an uniform way by translating the generating axioms of the export specification.

1. For the sort pre-E$_i$.

rs-E$_i$#(pent$i$-$_1$)
begin
  var a$_{i_1}$ = ?, ..., a$_{i_n}$ = ?, pent$i$-$_0$ = pent$i$-$_1$ **in**
  begin
    create-E$_i$#(a$_{i_1}$, ..., a$_{i_n}$; pent$i$-$_0$);
    if pent$i$-$_0$ = pent$i$-$_1$ **then skip else**
      var pent$_i$ = pent$i$-$_1$ **in**
      begin
        error-E$_i$#(;pent$_i$);
        if pent$_i$ = pent$i$-$_1$ **then skip else abort**
      end
  end
end

2. For the sort p-keysort-E_i.

   rs-key-E_i#(pent$i$-$_1$)
   begin
   var a_{i_k} = ?, ..., a_{i_l} = ?, $pkey_i$ = pkey$i$-$_1$ **in**
   begin
   mkkey-E_i#(a_{i_k}, ..., a_{i_l}; $pkey_i$);
   if $pkey_i$ = pkey$i$-$_1$ **then skip else abort**
   end
   end

The variable declarations with the question mark stand for nondeterministic initializations. Those initializations are not allowed in programs implementing function, but only in programs for restricting data. During a symbolic execution they are replaced by existential quantifiers.

At first we proof the generatedness of the entities and then the generatedness of the keys.

⟨rs-E_i#(pent$i$-$_1$)⟩ true ⊢ ⟨uniform_rs-E_i#(pent$i$-$_1$)⟩ true

Proof: At first we call the right side and then the left side. This yields two new goals:

1.

⊢

$\exists$ a_{i_1}, ..., a_{i_n}.

⟨**begin**
create-E_i#(a_{i_1}, ..., a_{i_n}; pent$i$-$_2$);
if pent$i$-$_2$ = p-error-E_i **then skip else**
var $pent_i$ = p-error-E_i **in**
begin
error-E_i#(;$pent_i$);
if $pent_i$ = p-error-E_i **then skip else abort**
end
end ⟩ true

2.

p-attr$_{i_k}$(pent$i$-$_1$) $\neq$ undef-Attr$_{i_k}$, ..., p-attr$_{i_l}$(pent$i$-$_1$)$\neq$ undef-Attr$_{i_l}$,

p-attr$_{i_1}$(pent$i$-$_1$) $\neq$ error-Attr$_{i_1}$, ..., p-attr$_{i_n}$(pent$i$-$_1$)$\neq$ error-Attr$_{i_n}$,

pent$i$-$_1$ $\neq$ p-error-E_i

⊢

$\exists$ a_{i_1}, ..., a_{i_n}.

⟨**begin**
create-E_i#(a_{i_1}, ..., a_{i_n}; pent$i$-$_2$);
if pent$i$-$_2$ = pent$i$-$_1$ **then skip else**
var $pent_i$ = pent$i$-$_1$ **in**
begin
error-E_i#(;$pent_i$);
if $pent_i$ = pent$i$-$_1$ **then skip else abort**
end
end⟩ true

We instantiate the variables in the first case with the *error-Attr* constants and close this goal by a call. For the second goal we instantiate the variables with *p-attr$_i$(penti-$_1$)*. Then we call the right side and close this goal too.

The generatedness of the keys.

⟨rs-key-E_i#(pkey$i$-$_1$)⟩ true ⊢ ⟨uniform_rs-key-E_i#(pkey$i$-$_1$)⟩ true

Proof: At first we call the right side, this yields:

> ⟨rs-key-E_i#(pkey$i$-$_1$)⟩ true
>
> ⊢
>
> ∃ a_{i_k}, ..., a_{i_l}.
>
> ⟨mkkey-E_i#(a_{i_k}, ..., a_{i_l}; pkey$i$-$_0$);
> **if** pkey$i$-$_0$ = pkey$i$-$_1$ **then skip else abort**⟩ true
>
> Then we instantiate the variables a_{i_j} with *k-Attr*$_{i_j}$*(pkeyi-*$_1$*)* and we unfold the right side repeatedly and close the goal.

This completes the proof for consistency of the specifications $Entity_1$ until $Entity_n$.

4.3 The Database

In this section we present the heart of our translation of E/R-diagrams, the database specification.

Entity relationship diagrams consist of entity types modeling possible sets of data and binary relations, the principal conjunctions between these sets. E/R-diagrams represent such a structure. Therefore it is possible to model an E/R-diagram through a relational database.

In principal, entity types are represented as sets of entities and the binary relations as sets of pairs of keys. The four different kinds of such relations are only of interest for static integrity conditions and not for the elementary database. So we can define the database as a tuple of sets of entities and sets of pairs of keys. Not each composition of these components is possible, because two properties must be satisfied by every database.

1. The identification of an entity must be unambiguous. This means we have to guarantee that the keys of two entities are pairwise different.
2. No relation exists between entities which are not member in the database.

To make work possible we have to add basic operations on the database for data access and modification:

ent-E: A function of type (database → sets of Entities). For selecting different entity sets.

put-E: A function of type ((entity type × database) → database). For putting entities into the database.

del-E: A function of type ((entity type × database) → database). For deleting entities in the database.

get-E: A function of type ((keys × database) → entity type). For selecting entities by their key.

update-E: A function of type ((keys × entity type × database) → database). For modifying an entity in the database — preserving the key.

est-R: A function of type ((database × entity type × entity type) → database). For establishing a relation between two entities.

rel-R: A function of type ((database × entity type × entity type) → database). For relegating a relation between two entities.

R: A predicate of type (database × entity type × entity type).For testing whether there is an established relation between two entities.

If we want to implement such a database we cannot use arbitrary sets. To guarantee the properties 1. and 2. above we must have the possibility of to enumerate the used sets. This is only possible when we use sets over ordered elements to select minimal elements by recursion. Therefore we introduced in the previous sections the specifications *coded-set* for representing the entity types in the database and *pair-orderset* for representing the binary relations between entity types. Now we can describe the database specification as an enrichment of the union of actualized *coded-sets* and *pair-ordersets*. Because KIV only has the possibility of enriching a single specification this is a step by step proceeding.
First, the union of two *Entity* specifications needed for the representation of relations:

$\text{Entity}_{r_{j1}}\text{\_u\_Entity}_{r_{j2}} = \text{Entity}_{r_{j1}} + \text{Entity}_{r_{j2}}$

Second, the actualizations of *coded-set*, modeling the entity types:

set-E_i =
actualize coded-set **with** E_i **by morphism**
coded-set → set-E_i, element → E_i, key → keysort-E_i, error-elem → error-E_i, encode → key-E_i, $\ll_{key}$ → $\ll_{key-E_i}$, an-elem → ent_i, a-key → key_i, a-key_1 → key$i$-$_1$, a-key_2 → key$i$-$_2$, empty-c-set → emptyset-E_i, error-c-set → errorset-E_i, $+_{cs}$ → $+_{E_i}$, -_{cs} → -_{E_i}, sel → sel-E_i, min-cs → min-E_i, rest-cs → rest-E_i, in-cs → in-E_i, realsub-cs → rsub-E_i, el → ent$i$-$_0$, el_1 → ent$i$-$_1$, el_2 → ent$i$-$_2$, cs → sent_i, cs_1 → sent$i$-$_1$, cs_2 → sent$i$-$_2$
end actualize

Third, the actualizations of *pair-orderset*, modeling the relations:

R_i =
actualize pair-orderset **with** $E_{r_{i1}}\text{\_u\_}E_{r_{i2}}$ **by morphism**
pair-oset → reltype-R_i, elem1 → keysort-$E_{r_{i1}}$, elem2 → keysort-$E_{r_{i2}}$, $\ll_{elem1}$ → $\ll_{key-E_{r_{i1}}}$, $\ll_{elem2}$ → $\ll_{key-E_{r_{i2}}}$, e_0 → ki-$_0$, e_1 → ki-$_1$, e1-$_1$ → ki-1-$_1$, e1-$_2$ → ki-1-$_2$, e_2 → ki-$_2$, e2-$_1$ → ki-2-$_1$, e2-$_2$ → ki-2-$_2$, e_3 → ki-$_3$, pair → pair-R_i, $\ll_{pair}$ → $\ll_{R_i}$, mkpair → mk-R_i, .fst → fst-R_i, .snd → snd-R_i, par_1 → ki-$_1$, par_2 → ki-$_2$, p → pair_i, p_1 → pair$i$-$_1$, p_2 → pair$i$-$_2$, empty-p-set → emptyrel-R_i, $+_{ps}$ → $+_{R_i}$, -_{ps} → -_{R_i}, min-ps → min-R_i, rest-ps → rest-R_i, in-ps → in-R_i, realsub-ps → rsub-R_i, ap → pair_i, ap_1 → pair$i$-$_1$, ap_2 → pair$i$-$_2$, ps → rel_i, ps_1 → rel$i$-$_1$, ps_2 → rel$i$-$_2$
end actualize

Fourth,the union of *set-Entity*:

ENTITIESETS = set-E_1 + . . . + set-E_n

Fifth, union of *R*:

RELATIONS = R_1 + . . . + R_m

Sixth, the union of *ENTITIESETS* and *RELATIONS*:

ESETS+REL = ENTITIESETS + RELATIONS

At last we can present the database specification:

ER_DB =
enrich ESETS+REL **with**
sorts db;
constants empty-db : db; error-db : db;
functions
mk-db : set-E_1 × . . .
× set-E_n
× reltype-R_1 × . . .
× reltype-R_m → db ;

$$
\begin{array}{llll}
\text{ent-E}_1 & : \ \text{db} & \rightarrow & \text{set-E}_1 \ ; \\
\vdots & & & \\
\text{ent-E}_n & : \ \text{db} & \rightarrow & \text{set-E}_n \ ; \\
\text{put-E}_1 & : \ \text{E}_1 \times \text{db} & \rightarrow & \text{db} \ ; \\
\vdots & & & \\
\text{put-E}_n & : \ \text{E}_n \times \text{db} & \rightarrow & \text{db} \ ; \\
\text{del-E}_1 & : \ \text{E}_1 \times \text{db} & \rightarrow & \text{db} \ ; \\
\vdots & & & \\
\text{del-E}_n & : \ \text{E}_n \times \text{db} & \rightarrow & \text{db} \ ; \\
\text{get-E}_1 & : \ \text{keysort-E}_1 \times \text{db} & \rightarrow & \text{E}_1 \ ; \\
\vdots & & & \\
\text{get-E}_n & : \ \text{keysort-E}_n \times \text{db} & \rightarrow & \text{E}_n \ ; \\
\text{update-E}_1 & : \ \text{keysort-E}_1 \times \text{E}_1 \times \text{db} & \rightarrow & \text{db} \ ; \\
\vdots & & & \\
\text{update-E}_n & : \ \text{keysort-E}_n \times \text{E}_n & & \\
 & \quad \times \ \text{db} & \rightarrow & \text{db} \ ; \\
\text{est-R}_1 & : \ \text{db} \times \text{E}_{r_{11}} \times \text{E}_{r_{12}} & \rightarrow & \text{db} \ ; \\
\vdots & & & \\
\text{est-R}_m & : \ \text{db} \times \text{E}_{r_{m1}} \times \text{E}_{r_{m2}} & \rightarrow & \text{db} \ ; \\
\text{rel-R}_1 & : \ \text{db} \times \text{E}_{r_{11}} \times \text{E}_{r_{12}} & \rightarrow & \text{db} \ ; \\
\vdots & & & \\
\text{rel-R}_m & : \ \text{db} \times \text{E}_{r_{m1}} \times \text{E}_{r_{m2}} & \rightarrow & \text{db} \ ;
\end{array}
$$

predicates

$$
\begin{array}{ll}
\text{R}_1 & : \ \text{db} \times \text{E}_{r_{11}} \times \text{E}_{r_{12}}; \\
\vdots & \\
\text{R}_m & : \ \text{db} \times \text{E}_{r_{m1}} \times \text{E}_{r_{m2}};
\end{array}
$$

variables vdb: db;

axioms

db **generated by** mk-db, error-db;

$\text{mk-db}(\text{sent}_1, \ldots, \text{sent}_n, \text{rel}_1, \ldots, \text{rel}_m) \neq \text{error-db}$

$\leftrightarrow$

property 1

$$
\begin{array}{l}
(\forall\ \text{ent1-}_1, \text{ent1-}_2.\ \text{ent1-}_1 \text{ in-E}_1 \text{ sent}_1 \wedge \text{ent1-}_2 \text{ in-E}_1 \text{ sent}_1 \wedge \text{ent1-}_1 \neq \text{ent1-}_2 \\
\qquad\qquad \rightarrow \text{key-E}_1(\text{ent1-}_1) \neq \text{key-E}_1(\text{ent1-}_2)) \wedge \\
\vdots \\
\wedge\ (\forall\ \text{ent}n\text{-}_1, \text{ent}n\text{-}_2.\ \text{ent}n\text{-}_1 \text{ in-E}_n \text{ sent}_n \wedge \text{ent}n\text{-}_2 \text{ in-E}_n \text{ sent}_n \wedge \text{ent}n\text{-}_1 \neq \text{ent}n\text{-}_2 \\
\qquad\qquad \rightarrow \text{key-E}_n(\text{ent}n\text{-}_1) \neq \text{key-E}_n(\text{ent}n\text{-}_2))
\end{array}
$$

property 2

$$
\begin{array}{l}
\wedge\ (\forall\ \text{k1-}_1, \text{k1-}_2.\ \text{mk-R}_1(\text{k1-}_1, \text{k1-}_2) \text{ in-R}_1 \text{ rel}_1 \\
\qquad\qquad \rightarrow (\exists\ \text{ent}r_{11}\text{-}_1, \text{ent}r_{12}\text{-}_2.\ \text{ent}r_{11}\text{-}_1 \text{ in-E}_{r_{11}} \text{ sent}_{r_{11}} \\
\qquad\qquad\qquad \wedge\ \text{ent}r_{12}\text{-}_2 \text{ in-E}_{r_{12}} \text{ sent}_{r_{12}} \\
\qquad\qquad\qquad \wedge\ \text{key-E}_{r_{11}}(\text{ent}r_{11}\text{-}_1) = \text{k1-}_1 \wedge \text{key-E}_{r_{12}}(\text{ent}r_{12}\text{-}_2) = \text{k1-}_2)) \wedge \\
\vdots \\
\wedge\ (\forall\ \text{k}m\text{-}_1, \text{k}m\text{-}_2.\ \text{mk-R}_m(\text{k}m\text{-}_1, \text{k}m\text{-}_2) \text{ in-R}_m \text{ rel}_m \\
\qquad\qquad \rightarrow (\exists\ \text{ent}r_{m1}\text{-}_1, \text{ent}r_{m2}\text{-}_2.\ \text{ent}r_{m1}\text{-}_1 \text{ in-E}_{r_{m1}} \text{ sent}_{r_{m1}} \\
\qquad\qquad\qquad \wedge\ \text{ent}r_{m2}\text{-}_2 \text{ in-E}_{r_{m2}} \text{ sent}_{r_{m2}} \\
\qquad\qquad\qquad \wedge\ \text{key-E}_{r_{m1}}(\text{ent}r_{m1}\text{-}_1) = \text{k}m\text{-}_1 \wedge \text{key-E}_{r_{m2}}(\text{ent}r_{m2}\text{-}_2) = \text{k}m\text{-}_2))
\end{array}
$$

$\wedge\ \text{sent}_1 \neq \text{errorset-E}_1\ \wedge$

$\vdots$

$\wedge\ \text{sent}_n \neq \text{errorset-E}_n,$

$\text{mk-db}(\text{sent}_1, \ldots, \text{sent}_n, \text{rel}_1, \ldots, \text{rel}_m) \neq \text{error-db}$

$\rightarrow (\text{mk-db}(\text{sent}_1, \ldots, \text{sent}_n, \text{rel}_1, \ldots, \text{rel}_m)$

$\quad = \text{mk-db}(\text{sent1-}_1, \ldots, \text{sent}n\text{-}_1,$

$\qquad\qquad \text{rel1-}_1, \ldots, \text{rel}m\text{-}_1)$

$\quad \rightarrow \text{sent}_1 = \text{sent1-}_1 \wedge \ldots \wedge \text{sent}_n = \text{sent}n\text{-}_1$

$\qquad \wedge\ \text{rel}_1 = \text{rel1-}_1 \wedge \ldots \wedge \text{rel}_m = \text{rel}m\text{-}_1),$

$\text{empty-db} = \text{mk-db}(\text{emptyset-E}_1, \ldots, \text{emptyset-E}_n,$

$\qquad\qquad\qquad \text{emptyrel-R}_1, \ldots, \text{emptyrel-R}_m),$

$\text{vdb} \neq \text{error-db}$

$\wedge\ \text{vdb} = \text{mk-db}(\text{sent}_1, \ldots, \text{sent}_n, \text{rel}_1, \ldots, \text{rel}_m)$

$\rightarrow \text{ent-E}_1(\text{vdb}) = \text{sent}_1\ \wedge$

$\quad\vdots$

$\quad \wedge\ \text{ent-E}_n(\text{vdb}) = \text{sent}_n$

$\quad \wedge\ \text{put-E}_1(\text{ent}_1, \text{vdb})$

$\qquad = \text{mk-db}(\text{sent}_1 +_{E_1} \text{ent}_1, \ldots,$

$\qquad\qquad \text{sent}_n, \text{rel}_1, \ldots, \text{rel}_m)\ \wedge$

$\quad\vdots$

$\quad \wedge\ \text{put-E}_n(\text{ent}_n, \text{vdb})$

$\qquad = \text{mk-db}(\text{sent}_1, \ldots,$

$\qquad\qquad \text{sent}_n +_{E_n} \text{ent}_n,$

$\qquad\qquad \text{rel}_1, \ldots, \text{rel}_m)$

$\quad \wedge\ \text{del-E}_1(\text{ent}_1, \text{vdb})$

$\qquad = \text{mk-db}(\text{sent}_1 -_{E_1} \text{ent}_1, \ldots,$

$\qquad\qquad \text{sent}_n, \text{rel}_1, \ldots, \text{rel}_m)\ \wedge$

$\quad\vdots$

$\quad \wedge\ \text{del-E}_n(\text{ent}_n, \text{vdb})$

$\qquad = \text{mk-db}(\text{sent}_1, \ldots,$

$\qquad\qquad \text{sent}_n -_{E_n} \text{ent}_n,$

$\qquad\qquad \text{rel}_1, \ldots, \text{rel}_m),$

$\text{get-E}_1(\text{key}_1, \text{vdb}) \neq \text{error-E}_1$

$\leftrightarrow (\exists\ \text{ent}_1.\text{ent}_1\ \text{in-E}_1\ \text{ent-E}_1(\text{vdb}) \wedge \text{key-E}_1(\text{ent}_1) = \text{key}_1),$

$\vdots$

$\text{get-E}_n(\text{key}_n, \text{vdb}) \neq \text{error-E}_n$

$\leftrightarrow (\exists\ \text{ent}_n.\text{ent}_n\ \text{in-E}_n\ \text{ent-E}_n(\text{vdb}) \wedge \text{key-E}_n(\text{ent}_n) = \text{key}_n),$

$\text{ent}_1 \neq \text{error-E}_1$

$\rightarrow (\text{get-E}_1(\text{key}_1, \text{vdb}) = \text{ent}_1$

$\quad \leftrightarrow \text{ent}_1\ \text{in-E}_1\ \text{ent-E}_1(\text{vdb}) \wedge \text{key-E}_1(\text{ent}_1) = \text{key}_1),$

$\vdots$

$\text{ent}_n \neq \text{error-E}_n$

$\rightarrow (\text{get-E}_n(\text{key}_n, \text{vdb}) = \text{ent}_n$

$\quad \leftrightarrow \text{ent}_n\ \text{in-E}_n\ \text{ent-E}_n(\text{vdb}) \wedge \text{key-E}_n(\text{ent}_n) = \text{key}_n),$

$\text{update-E}_1(\text{key}_1, \text{ent}_1, \text{vdb}) \neq \text{error-db}$

$\leftrightarrow (\exists\ \text{ent1-}_2.\text{ent1-}_2\ \text{in-E}_1\ \text{ent-E}_1(\text{vdb})$

$\qquad\qquad \wedge\ \text{key-E}_1(\text{ent1-}_2) = \text{key}_1$

$\qquad\qquad \wedge\ \text{key-E}_1(\text{ent}_1) = \text{key}_1),$

$\vdots$

$\text{update-E}_n(\text{key}_n, \text{ent}_n, \text{vdb}) \neq \text{error-db}$

$\leftrightarrow (\exists\ \text{ent}n\text{-}_2.\text{ent}n\text{-}_2\ \text{in-E}_n\ \text{ent-E}_n(\text{vdb})$

$\qquad\qquad \wedge\ \text{key-E}_n(\text{ent}n\text{-}_2) = \text{key}_n$

$$
\begin{aligned}
&\qquad\qquad \wedge\ \text{key-E}_n(\text{ent}_n) = \text{key}_n),\\
&\text{update-E}_1(\text{key}_1, \text{ent}_1, \text{vdb}) \neq \text{error-db}\\
&\wedge\ \text{vdb} = \text{mk-db}(\text{sent}_1, \ldots, \text{sent}_n, \text{rel}_1, \ldots, \text{rel}_m)\\
&\rightarrow \text{update-E}_1(\text{key}_1, \text{ent}_1, \text{vdb})\\
&\quad = \text{mk-db}(\text{sent}_1 \text{ -}_{E_1} \text{ get-E}_1(\text{key}_1, \text{vdb}) +_{E_1} \text{ent}_1, \ldots,\\
&\qquad\qquad \text{sent}_n, \text{rel}_1, \ldots, \text{rel}_m),\\
&\vdots\\
&\text{update-E}_n(\text{key}_n, \text{ent}_n, \text{vdb}) \neq \text{error-db}\\
&\wedge\ \text{vdb} = \text{mk-db}(\text{sent}_1, \ldots, \text{sent}_n, \text{rel}_1, \ldots, \text{rel}_m)\\
&\rightarrow \text{update-E}_n(\text{key}_n, \text{ent}_n, \text{vdb})\\
&\quad = \text{mk-db}(\text{sent}_1, \ldots,\\
&\qquad\qquad \text{sent}_n \text{ -}_{E_n} \text{ get-E}_n(\text{key}_n, \text{vdb}) +_{E_n} \text{ent}_n,\\
&\qquad\qquad \text{rel}_1, \ldots, \text{rel}_m),\\
&\text{vdb} \neq \text{error-db}\\
&\wedge\ \text{vdb} = \text{mk-db}(\text{sent}_1, \ldots, \text{sent}_n, \text{rel}_1, \ldots, \text{rel}_m)\\
&\wedge\ \text{ent}_{r_{11}} \neq \text{error-E}_{r_{11}}\\
&\wedge\ \text{ent}_{r_{12}} \neq \text{error-E}_{r_{12}}\\
&\rightarrow (\text{R}_1(\text{vdb}, \text{ent}_{r_{11}}, \text{ent}_{r_{12}})\\
&\leftrightarrow\\
&\text{mk-R}_1(\text{key-E}_{r_{11}}(\text{ent}_{r_{11}}), \text{key-E}_{r_{12}}(\text{ent}_{r_{12}})) \text{ in-R}_1 \text{ rel}_1)\\
&\wedge\ \text{est-R}_1(\text{vdb}, \text{ent}_{r_{11}}, \text{ent}_{r_{12}})\\
&\quad = \text{mk-db}(\text{sent}_1, \ldots, \text{sent}_n,\\
&\qquad\qquad \text{rel}_1 +_{R_1} \text{mk-R}_1(\text{key-E}_{r_{11}}(\text{ent}_{r_{11}}),\\
&\qquad\qquad\qquad\qquad \text{key-E}_{r_{12}}(\text{ent}_{r_{12}})), \ldots,\\
&\qquad\qquad \text{rel}_m)\\
&\wedge\ \text{rel-R}_1(\text{vdb}, \text{ent}_{r_{11}}, \text{ent}_{r_{12}})\\
&\quad = \text{mk-db}(\text{sent}_1, \ldots, \text{sent}_n,\\
&\qquad\qquad \text{rel}_1 \text{ -}_{R_1} \text{ mk-R}_1(\text{key-E}_{r_{11}}(\text{ent}_{r_{11}}),\\
&\qquad\qquad\qquad\qquad \text{key-E}_{r_{12}}(\text{ent}_{r_{12}})), \ldots,\\
&\qquad\qquad \text{rel}_m),\\
&\vdots\\
&\text{vdb} \neq \text{error-db}\\
&\wedge\ \text{vdb} = \text{mk-db}(\text{sent}_1, \ldots, \text{sent}_n, \text{rel}_1, \ldots, \text{rel}_m)\\
&\wedge\ \text{ent}_{r_{m1}} \neq \text{error-E}_{r_{m1}}\\
&\wedge\ \text{ent}_{r_{m2}} \neq \text{error-E}_{r_{m2}}\\
&\rightarrow (\text{R}_m(\text{vdb}, \text{ent}_{r_{m1}}, \text{ent}_{r_{m2}})\\
&\leftrightarrow\\
&\text{mk-R}_m(\text{key-E}_{r_{m1}}(\text{ent}_{r_{m1}}), \text{key-E}_{r_{m2}}(\text{ent}_{r_{m2}})) \text{ in-R}_m \text{ rel}_m)\\
&\wedge\ \text{est-R}_m(\text{vdb}, \text{ent}_{r_{m1}}, \text{ent}_{r_{m2}})\\
&\quad = \text{mk-db}(\text{sent}_1, \ldots, \text{sent}_n, \text{rel}_1, \ldots,\\
&\qquad\qquad \text{rel}_m +_{R_m} \text{mk-R}_m(\text{key-E}_{r_{m1}}(\text{ent}_{r_{m1}}),\\
&\qquad\qquad\qquad\qquad \text{key-E}_{r_{m2}}(\text{ent}_{r_{m2}})))\\
&\wedge\ \text{rel-R}_m(\text{vdb}, \text{ent}_{r_{m1}}, \text{ent}_{r_{m2}})\\
&\quad = \text{mk-db}(\text{sent}_1, \ldots, \text{sent}_n, \text{rel}_1, \ldots,\\
&\qquad\qquad \text{rel}_m \text{ -}_{R_m} \text{ mk-R}_m(\text{key-E}_{r_{m1}}(\text{ent}_{r_{m1}}),\\
&\qquad\qquad\qquad\qquad \text{key-E}_{r_{m2}}(\text{ent}_{r_{m2}}))),\\
&\text{ent-E}_1(\text{error-db}) = \text{errorset-E}_1,\\
&\vdots\\
&\text{ent-E}_n(\text{error-db}) = \text{errorset-E}_n,\\
&\text{put-E}_1(\text{ent}_1, \text{error-db}) = \text{error-db},\\
&\vdots\\
&\text{put-E}_n(\text{ent}_n, \text{error-db}) = \text{error-db},\\
&\text{del-E}_1(\text{ent}_1, \text{error-db}) = \text{error-db},
\end{aligned}
$$

⋮
del-E_n(ent$_n$, error-db) = error-db,
vdb = error-db ∨ ent$_{r_{11}}$ = error-$E_{r_{11}}$ ∨ ent$_{r_{12}}$ = error-$E_{r_{12}}$
→ ¬ R_1(vdb, ent$_{r_{11}}$, ent$_{r_{12}}$)
∧ est-R_1(vdb, ent$_{r_{11}}$, ent$_{r_{12}}$) = error-db
∧ rel-R_1(vdb, ent$_{r_{11}}$, ent$_{r_{12}}$) = error-db,
⋮
vdb = error-db ∨ ent$_{r_{m1}}$ = error-$E_{r_{m1}}$ ∨ ent$_{r_{m2}}$ = error-$E_{r_{m2}}$
→ ¬ R_m(vdb, ent$_{r_{m1}}$, ent$_{r_{m2}}$)
∧ est-R_m(vdb, ent$_{r_{m1}}$, ent$_{r_{m2}}$) = error-db
∧ rel-R_m(vdb, ent$_{r_{m1}}$, ent$_{r_{m2}}$) = error-db
end enrich

4.3.1 The Implementation

To guarantee the consistency of this enrichment we implement the database specification *ER_DB*. As a basis for our implementation we use the specification *pre-DB*, which establishes the sort *pre-db* as carrier set of a freely generated algebra over the function *p-mk-db* and an error constant *p-error-db*.

pre-DB_herzkatheter =
data specification
using ESETS+REL_herzkatheter
pre-db = p-mk-db (p-ent-E_1 : set-E_1, ...,
p-ent-E_n : set-E_n,
p-R_1 : reltype-R_1, ...,
p-R_m : reltype-R_m)
| p-error-db
;
variables vpdb: pre-db;
end data specification

In the implementation of *ER_DB* the sort *db* is represented by a subset of *pre-db* containing only the terms *p-error-db* and *p-mk-db(...)* where both properties are fulfilled (see above).

The module:

DB-preDB =
module
export ER_DB_herzkatheter
refinement
representation of sorts
pre-db implements db;
representation of operations
empty-db# implements empty-db;
error-db# implements error-db;
mk-db# implements mk-db;

ent-E_1#	implements	ent-E_1;
$\vdots$		
ent-E_n#	implements	ent-E_n;
put-E_1#	implements	put-E_1;
$\vdots$		
put-E_n#	implements	put-E_n;
del-E_1#	implements	del-E_1;
$\vdots$		
del-E_n#	implements	del-E_n;
get-E_1#	implements	get-E_1;
$\vdots$		
get-E_n#	implements	get-E_n;
update-E_1#	implements	update-E_1;
$\vdots$		
update-E_n#	implements	update-E_n;
est-R_1#	implements	est-R_1;
$\vdots$		
est-R_m#	implements	est-R_m;
rel-R_1#	implements	rel-R_1;
$\vdots$		
rel-R_m#	implements	rel-R_m;
R_1#	implements	R_1;
$\vdots$		
R_m#	implements	R_m;

import pre-DB_herzkatheter

procedures

empty-db#	()	: pre-db;
error-db#	()	: pre-db;
mk-db#	(set-E_1, ..., set-E_n,	
	reltype-R_1, ..., reltype-R_m)	: pre-db;
ent-E_1#	(pre-db)	: set-E_1;
$\vdots$		
ent-E_n#	(pre-db)	: set-E_n;
put-E_1#	(E_1, pre-db)	: pre-db;
$\vdots$		
put-E_n#	(E_n, pre-db)	: pre-db;
del-E_1#	(E_1, pre-db)	: pre-db;
$\vdots$		
del-E_n#	(E_n, pre-db)	: pre-db;
get-E_1#	(keysort-E_1, pre-db)	: E_1;
$\vdots$		
get-E_n#	(keysort-E_n, pre-db)	: E_n;
update-E_1#	(keysort-E_1, E_1, pre-db)	: pre-db;
$\vdots$		
update-E_n#	(keysort-E_n, E_n, pre-db)	: pre-db;

est-R_1#	(pre-db, E_{r11}, E_{r12})	: pre-db;
⋮		
est-R_m#	(pre-db, E_{rm1}, E_{rm2})	: pre-db;
rel-R_1#	(pre-db, E_{r11}, E_{r12})	: pre-db;
⋮		
rel-R_m#	(pre-db, E_{rm1}, E_{rm2})	: pre-db;
R_1#	(pre-db, E_{r11}, E_{r12})	: bool;
⋮		
R_m#	(pre-db, E_{rm1}, E_{rm2})	: bool;

Test functions for property 2. Property 1 still holds according to the used set version.

legal-R_1#	(reltype-R_1, set-E_{r11}, set-E_{r12})	: bool;
⋮		
legal-R_m#	(reltype-R_m, set-E_{rm1}, set-E_{rm2})	: bool;
in-fst-R_1#	(keysort-E_{r11}, reltype-R_1)	: bool;
⋮		
in-fst-R_m#	(keysort-E_{rm1}, reltype-R_m)	: bool;
in-snd-R_1#	(keysort-E_{r12}, reltype-R_1)	: bool;
⋮		
in-snd-R_m#	(keysort-E_{rm2}, reltype-R_m)	: bool;

variables pdb_1, pdb: pre-db; b: bool;

implementation

empty-db#(**var** pdb)
begin
  pdb := p-mk-db(emptyset-E_1, ...,
    emptyset-E_n,
    emptyrel-R_1, ...,
    emptyrel-R_m)
end

error-db#(**var** pdb) **begin** pdb := p-error-db **end**

mk-db#($sent_1$, ..., $sent_n$, rel_1, ..., rel_m; **var** pdb)
begin
  if $sent_1$ = errorset-E_1 ∨
    ⋮
    ∨ $sent_n$ = errorset-E_n **then**
   pdb := p-error-db
  else
   var b = tt **in**

```
    begin
      legal-R$_1$#(rel$_1$, sent$_{r_{11}}$, sent$_{r_{12}}$;b); Test of property 2.
      if b = ff then pdb := p-error-db else
        begin
          ⋮
          if b = ff then pdb := p-error-db else
            begin
              legal-R$_m$#(rel$_m$, sent$_{r_{m1}}$, sent$_{r_{m2}}$;b);
              Test of property 2
              if b = ff then pdb := p-error-db else
                pdb := p-mk-db(sent$_1$, ..., sent$_n$,
                               rel$_1$, ..., rel$_m$)
            end
        end
    end
end


ent-E$_1$#(pdb; var sent$_1$)
begin
  if pdb = p-error-db then sent$_1$ := errorset-E$_1$ else
    sent$_1$ := p-ent-E$_1$(pdb)
end

  ⋮

ent-E$_n$#(pdb; var sent$_n$)
begin
  if pdb = p-error-db then sent$_n$ := errorset-E$_n$ else
    sent$_n$ := p-ent-E$_n$(pdb)
end


put-E$_1$#(ent$_1$, pdb; var pdb$_1$)
begin
  if pdb = p-error-db then pdb$_1$ := p-error-db else
    var sent$_1$ = p-ent-E$_1$(pdb) +$_{E_1}$ ent$_1$ in
    +$_{E_1}$ includes test of property 1 implicit.
    if sent$_1$ = errorset-E$_1$ then pdb$_1$ := p-error-db else
      pdb$_1$ := p-mk-db(sent$_1$, ...
                        p-ent-E$_n$(pdb),
                        p-R$_1$(pdb), ...
                        p-R$_m$(pdb))
end

  ⋮

put-E$_n$#(ent$_n$, pdb; var pdb$_1$)
begin
  if pdb = p-error-db then pdb$_1$ := p-error-db else
    var sent$_n$ = p-ent-E$_n$(pdb) +$_{E_n}$ ent$_n$ in
    +$_{E_n}$ includes test of property 1 implicit.
```

```
   if sent_n = errorset-E_n then pdb_1 := p-error-db else
     pdb_1 := p-mk-db(p-ent-E_1(pdb), ...
                        sent_n,
                        p-R_1(pdb), ...
                        p-R_m(pdb))
end


del-E_1#(ent_1, pdb; var pdb_1)
begin
  if ent_1 = error-E_1 ∨ pdb = p-error-db then pdb_1 := p-error-db else
    var key_1 = key-E_1(ent_1), b = tt in
    begin
      in-fst-R_i#(key_1, p-R_i(pdb);b); Test of property 2
      if b = tt then pdb_1 := p-error-db else
        begin
          in-fst- ...; Test of property 2
          if b = tt then pdb_1 := p-error-db else
            ⋮
            begin
              in-snd- ...; Test of property 2
              if b = tt then pdb_1 := p-error-db else
                var sent_1 = p-ent-E_1(pdb) -_E1 ent_1 in
                pdb_1 := p-mk-db(sent_1, ...,
                                   p-ent-E_n(pdb),
                                   p-R_1(pdb),
                                   p-R_m(pdb))
            end
        end
    end
end

   ⋮

del-E_n#(ent_n, pdb; var pdb_1)
begin
  if ent_n = error-E_n ∨ pdb = p-error-db then pdb_1 := p-error-db else
    var key_n = key-E_n(ent_n), b = tt in
    begin
      in-fst-R_j#(key_n, p-R_j(pdb);b); Test of property 2
      if b = tt then pdb_1 := p-error-db else
        begin
          in-fst- ...; Test of property 2
          if b = tt then pdb_1 := p-error-db else
            ⋮
            begin
              in-snd- ...; Test of property 2
              if b = tt then pdb_1 := p-error-db else
                var sent_n = p-ent-E_n(pdb) -_En entn in
                pdb_1 := p-mk-db(p-ent-E_1(pdb), ...,
                                   sent_n,
                                   p-R_1(pdb), ...,
                                   p-R_m(pdb))
```

```
                end
            end
        end
    end

get-E1#(key1, pdb; var ent1)
begin
    if pdb = p-error-db then ent1 := error-E1 else
        ent1 := sel-E1(key1, p-ent-E1(pdb))
end

    ⋮

get-En#(keyn, pdb; var entn)
begin
    if pdb = p-error-db then entn := error-En else
        entn := sel-En(keyn, p-ent-En(pdb))
end

update-E1#(key1, ent1, pdb; var pdb1)
begin
    if key-E1(ent1) ≠ key1 ∨ ent1 = error-E1 then
        pdb1 := p-error-db
    else
        var ent1-1 = error-E1 in
        begin
            get-E1#(key1, pdb;ent1-1);
            if ent1-1 = error-E1 then pdb1 := p-error-db else
                var sent1 = p-ent-E1(pdb) -E1 ent1-1 +E1 ent1 in
                pdb1 := p-mk-db(sent1, ...,
                                p-ent-En(pdb),
                                p-R1(pdb), ...,
                                p-Rm(pdb))
        end
end

    ⋮

update-En#(keyn, entn, pdb; var pdb1)
begin
    if key-En(entn) ≠ keyn ∨ entn = error-En then
        pdb1 := p-error-db
    else
        var entn-1 = error-En in
        begin
            get-En#(keyn, pdb;entn-1);
            if entn-1 = error-En then pdb1 := p-error-db else
                var sentn = p-ent-En(pdb) -En entn-1 +En entn in
                pdb1 := p-mk-db(p-ent-E1(pdb), ...,
                                sentn,
                                p-R1(pdb), ...,
                                p-Rm(pdb))
        end
end
```

est-R_1#(pdb, entr_{11}-$_1$, entr_{12}-$_2$; **var** pdb_1)
begin
  if pdb = p-error-db $\vee$ entr_{11}-$_1$ = error-$E_{r_{11}}$ $\vee$ entr_{12}-$_2$ = error-$E_{r_{12}}$ **then**
    pdb_1 := p-error-db
  else
    if sel-$E_{r_{11}}$(key-$E_{r_{11}}$(entr_{11}-$_1$), p-ent-$E_{r_{11}}$(pdb)) = error-$E_{r_{11}}$ *Test of property 2*
      $\vee$ sel-$E_{r_{12}}$(key-$E_{r_{12}}$(entr_{12}-$_2$),
        p-ent-$E_{r_{12}}$(pdb)) = error-$E_{r_{12}}$ **then**
      pdb_1 := p-error-db
    else
      var rel_1 = p-R_1(pdb)
        $+_{r_1}$ mk-R_1(key-$E_{r_{11}}$(entr_{11}-$_1$),
          key-$E_{r_{12}}$(entr_{12}-$_2$)) **in**
      pdb_1 := p-mk-db(p-ent-E_1(pdb), ...,
        p-ent-E_n(pdb),
        rel_1, ...,
        p-R_m(pdb))
end

⋮

est-R_m#(pdb, enr_{m1}-$_1$, enr_{m2}-$_2$; **var** pdb_1)
begin
  if pdb = p-error-db $\vee$ entr_{m1}-$_1$ = error-$E_{r_{m1}}$ $\vee$ entr_{m2}-$_2$ = error-$E_{r_{m2}}$ **then**
    pdb_1 := p-error-db
  else
    if sel-$E_{r_{m1}}$(key-$E_{r_{m1}}$(enr_{m1}-$_1$), p-ent-$E_{r_{m1}}$(pdb)) = error-$E_{r_{m1}}$ *Test of property 2*
      $\vee$ sel-$E_{r_{m2}}$(key-$E_{r_{m2}}$(enr_{m2}-$_2$),
        p-ent-$E_{r_{m2}}$(pdb)) = error-$E_{r_{m2}}$ **then**
      pdb_1 := p-error-db
    else
      var rel_m = p-R_m(pdb)
        $+_{R_m}$ mk-R_m(key-$E_{r_{m1}}$(enr_{m1}-$_1$),
          key-$E_{r_{m2}}$(enr_{m2}-$_2$)) **in**
      pdb_1 := p-mk-db(p-ent-E_1(pdb), ...,
        p-ent-E_n(pdb),
        p-R_1(pdb), ...,
        rel_m)
end

rel-R_1#(pdb, entr_{11}-$_1$, entr_{12}-$_2$; **var** pdb_1)
begin
  if pdb = p-error-db $\vee$ entr_{11}-$_1$ = error-$E_{r_{11}}$ $\vee$ entr_{12}-$_2$ = error-$E_{r_{12}}$ **then**
    pdb_1 := p-error-db
  else
    var rel_1 = p-R_1(pdb)
      $-_{r_1}$ mk-R_1(key-$E_{r_{11}}$(entr_{11}-$_1$),
        key-hk_ao(entr_{12}-$_2$)) **in**
    pdb_1 := p-mk-db(p-ent-E_1(pdb), ...,
      p-ent-E_n(pdb),
      rel_1, ...,
      p-R_m(pdb))
end

⋮

rel-R_m#(pdb, entr_{m1}-$_1$, entr_{m2}-$_2$; **var** pdb_1)
begin
  if pdb = p-error-db ∨ entr_{m1}-$_1$ = error-$E_{r_{m1}}$ ∨ entr_{m2}-$_2$ = error-$E_{r_{m2}}$ **then**
    pdb_1 := p-error-db
  else
    var rel_m = p-R_m(pdb)
                -$_{r_m}$ mk-R_m(key-$E_{r_{m1}}$(entr_{m1}-$_1$),
                        key-$E_{r_{m2}}$(entr_{m2}-$_2$)) **in**
    pdb_1 := p-mk-db(p-ent-E_1(pdb), . . .,
                p-ent-E_n(pdb),
                p-R_1(pdb), . . .,
                rel_m)
end

R_1#(pdb, entr_{11}-$_1$, entr_{12}-$_2$; **var** b)
begin
  if pdb = p-error-db ∨ entr_{11}-$_1$ = error-$E_{r_{11}}$ ∨ entr_{12}-$_2$ = error-$E_{r_{12}}$**then**
    b := ff
  else
    if mk-R_1(key-$E_{r_{11}}$(entr_{11}-$_1$), key-$E_{r_{12}}$(entr_{12}-$_2$))
      in-R_1 p-R_1(pdb) **then**
      b := tt
    else
      b := ff
end

⋮

R_m#(pdb, entr_{m1}-$_1$, entr_{m2}-$_2$; **var** b)
begin
  if pdb = p-error-db ∨ entr_{m1}-$_1$ = error-$E_{r_{m1}}$ ∨ entr_{m2}-$_2$ = error-$E_{r_{m2}}$ **then**
    b := ff
  else
    if mk-R_m(key-$E_{r_{m1}}$(entr_{m1}-$_1$), key-$E_{r_{m2}}$(entr_{m2}-$_2$))
      in-R_m p-R_m(pdb) **then**
      b := tt
    else
      b := ff
end

Test of property 2

legal-R_1#(rel_1, $sent_{r_{11}}$, $sent_{r_{12}}$; **var** b)
begin
  if rel_1 = emptyrel-R_1 **then** b := tt **else**
    var $pair_1$ = min-R_1(rel_1) **in**
    if sel-$E_{r_{11}}$(fst-R_1($pair_1$), $sent_{r_{11}}$) = error-$E_{r_{11}}$
      ∨ sel-$E_{r_{12}}$(snd-R_1($pair_1$), $sent_{r_{12}}$) = error-$E_{r_{12}}$ **then**
      b := ff
    else
      legal-R_1#(rest-R_1(rel_1), $sent_{r_{11}}$, $sent_{r_{12}}$;b)
end

⋮

```
legal-Rm#(relm, sentrm1, sentrm2; var b)
begin
  if relm = emptyrel-Rm then b := tt else
    var pairm = min-Rm(relm) in
    if sel-Erm1(fst-Rm(pairm), sentrm1) = error-Erm1
       ∨ sel-Erm2(snd-Rm(pairm), sentrm2) = error-Erm2 then
      b := ff
    else
      legal-Rm#(rest-Rm(relm), sentrm1, sentrm2;b)
end
```

Tests whether the key is in the first part of a relation, used for guaranteeing property 2 during the deletion of entities.

```
in-fst-R1#(kr11-1, rel1; var b)
begin
  if rel1 = emptyrel-R1 then b := ff else
    if kr11-1 = fst-R1(min-R1(rel1)) then b := tt else
      in-fst-R1#(kr11-1, rest-R1(rel1);b)
end
```

⋮

```
in-fst-Rm#(krm1-1, relm; var b)
begin
  if relm = emptyrel-Rm then b := ff else
    if krm1-1 = fst-Rm(min-Rm(relm)) then b := tt else
      in-fst-Rm#(krm1-1, rest-Rm(relm);b)
end
```

Tests whether the key is in the second part of a relation, used for guaranteeing property 2 during the deletion of entities.

```
in-snd-R1#(kr12-2, rel1; var b)
begin
  if rel1 = emptyrel-R1 then b := ff else
    if kr12-2 = snd-R1(min-R1(rel1)) then b := tt else
      in-snd-R1#(kr12-2, rest-R1(rel1);b)
end
```

⋮

```
in-snd-Rm#(krm2-2, relm; var b)
begin
  if relm = emptyrel-Rm then b := ff else
    if krm2-2 = snd-Rm(min-Rm(relm)) then b := tt else
      in-snd-Rm#(krm2-2, rest-Rm(relm);b)
end
```

4.3.2 Proof Obligations

In this section we present the proof obligations that guarantee the correctness of the implementation and so far the consistency of the specification *ER_DB*. Because we are now firm in proving we present the proofs in this section in a more sketched way. This is necessary to concentrate our view to central points. Before we are starting we present the restriction for modeling the used subset of *pre-db*. Like the restriction in the entity case this can be realized through a partial procedure.

restriction

```
rs#(pdb)
begin
  if pdb = p-error-db then skip else
   var pdb1 = p-error-db in
   begin
     mk-db#(p-ent-E1(pdb), ...,
            p-ent-En(pdb),
            p-R1(pdb), ...,
            p-Rm(pdb);
            pdb1);
     if pdb1 = p-error-db then abort
   end
end
```

Conditions for Terminating (i)

Even like above, we formulate a proof obligation for each function and constant of the specification *ER_DB* which guarantee the termination with respect to the restrictions.

1. Termination of *empty-db#*

   $\vdash \langle \text{empty-db\#}(;\text{pdb})\rangle\ \langle \text{rs\#}(\text{pdb})\rangle\ \text{true}$

   Proof: This goal can be proved by trivial unfold steps.

2. Termination of *error-db#*

   $\vdash \langle \text{error-db\#}(;\text{pdb})\rangle\ \langle \text{rs\#}(\text{pdb})\rangle\ \text{true}$

   Proof: This goal can be proved by trivial unfold steps too.

3. Termination of *mk-db#*

   $\vdash \langle \text{mk-db\#}(\text{sent}_1, \ldots, \text{sent}_n, \text{rel}_1, \ldots, \text{rel}_m;\text{pdb})\rangle\ \langle \text{rs\#}(\text{pdb})\rangle\ \text{true}$

   Proof: We unfold the procedure call of *mk-db#*. In each case if one the procedures *legal-...#* is called we insert one of the following lemmas:

   $$\vdash \langle \text{legal-R}_i\#(\text{rel}_i, \text{sent}_{r_{i1}}, \text{sent}_{r_{i2}};\ \text{b})\rangle\ \text{true}\ (1 \le i \le m)$$

   To prove such a lemma we use Noetherian induction over rel_i. As a well founded order we use the *realsub* relation of the specification *orderset*. In the base case, if rel_i is empty, the procedure terminates. In the other case we unfold the call. Because the rel_i is not empty we are working in the else case and make another case distinction. In the positive case the call terminates in the negative case there is a recursive call. Because not empty rel_i implies *rest-*$R_i(rel_i)$ *realsub* rel_i we can apply the induction hypothesis and finish the proof of this lemma.

After applying such lemmas we make a case distinction. In each positive case the procedure terminates with the error constant. Repeated unfold steps on the diamond $\langle rs\#(p\text{-}error\text{-}db)\rangle$ *true* close these sub goals. The result is one open goal:

$\langle$legal-$R_1\#(rel_1, sent_{r_{11}}, sent_{r_{12}}; b)\rangle$ b = tt,
⋮
$\langle$legal-$R_m\#(rel_m, sent_{r_{m1}}, sent_{r_{m2}}; b)\rangle$ b = tt,
$sent_1 \neq$ errorset-E_1, ..., $sent_n \neq$ errorset-E_n
⊢
$\langle$rs#(p-mk-db($sent_1$, ..., $sent_n$,
rel_1, ..., rel_m;pdb)$\rangle$ true

This goal can now be trivially closed by executing the right side.

4. Termination of *ent-E_i#*

$\langle$rs#(pdb)$\rangle$ true ⊢ $\langle$ent-E_i#(pdb;$sent_i$)$\rangle$ true

Proof: This goal can be proved by trivial unfold steps.

5. Termination of *put-E_i#*

$\langle$rs#(pdb)$\rangle$ true ⊢ $\langle$put-E_i#(ent_i, pdb;pdb_1)$\rangle$ $\langle$rs#(pdb_1)$\rangle$ true

Proof: To prove this goal we unfold the right side. The branches dealing with error cases can be closed in a trivial way. So the following open goal remains:

$\langle$rs#(pdb)$\rangle$ true,
pdb $\neq$ p-error-db, p-ent-E_i(pdb) $+_{E_i}$ $ent_i \neq$ errorset-E_i,
⊢
$\langle$rs#(p-mk-db(p-ent-E_1(pdb), ...,
p-ent-E_i(pdb) $+_{E_i}$ ent_i, ...,
p-ent-E_n(pdb),
p-R_1(pdb), ..., p-R_m(pdb)))$\rangle$ true

Then we call the left side. The error branches can be closed in a trivial way once more. We call the right side. All *legal-...#* calls can be eliminated until those which deal with relations containing the entity type E_i. For those we use the following lemmas. And we finish the proof.

1.
$\langle$legal-R_j#(rel_j, $sent_i$, $sent_{r_{j2}}$; b)$\rangle$ b = tt,
$sent_i$ $+_{E_i}$ $ent_i \neq$ errorset-E_i
⊢
$\langle$legal-R_j#(rel_j, $sent_i$ $+_{E_i}$ ent_i, $sent_{r_{j2}}$;b)$\rangle$ b = tt

2.
$\langle$legal-R_k#(rel_k, $sent_{r_{k1}}$, $sent_i$;b)$\rangle$ b = tt,
$sent_i$ $+_{E_i}$ $ent_i \neq$ errorset-E_i
⊢
$\langle$legal-R_k#(rel_m, $sent_{r_{k1}}$, $sent_i$ $+_{E_i}$ ent_i;b)$\rangle$ b = tt

To prove these lemmas we use Noetherian induction over rel_j (rel_k). As a well founded order we use the *realsub* relation of the specification *orderset*. In the base case, if rel_j is empty, both calls yield *tt*. In the other case we unfold the left side call and then the right side diamond, all the if statements of the right side can be decided. Then we can apply the induction hypothesis because not empty *rel_j* implies *rest-R_j(rel_j) realsub rel_j* and we finish the proof of these lemmas.

6. Termination of *del-E_i#*

$\langle$rs#(pdb)$\rangle$ true $\vdash$ $\langle$del-E_i#(ent_i, pdb;pdb_1)$\rangle$ $\langle$rs#(pdb_1)$\rangle$ true

Proof: Before we are going to prove this goal we need lemmas to guarantee the termination of the procedures *in-fst-...#* and *in-snd-...#*.

$$\vdash \langle \text{in-fst-R}_j\#(\text{k}_i, \text{rel}_j; \text{b})\rangle \text{ true}$$

$$\vdash \langle \text{in-snd-R}_k\#(\text{k}_i, \text{rel}_k; \text{b})\rangle \text{ true}$$

In an analogous way to the *legal-...#* procedures this can be done by induction over *rel_j (rel_k)*.
We start the proof with unfolding the right side. The branches dealing with error cases can be closed in a trivial way. So the following open goal remains:

$\langle$rs#(pdb)$\rangle$ true,
⋮
$\langle$in-fst-R_j#(key-E_i(ent_i),p-R_j(pdb); b_0)$\rangle$ b_0 = ff,
⋮
$\langle$in-snd-R_k#(key-E_i(ent_i), p-R_k(pdb); b_0)$\rangle$ b_0 = ff,
⋮
pdb $\neq$ p-error-db, ent_i $\neq$ error-E_i,
$\vdash$
$\langle$rs#(p-mk-db(p-ent-E_1(pdb), ...,
p-ent-E_i(pdb) $-_{E_i}$ ent_i, ...,
p-ent-E_n(pdb),
p-R_1(pdb), ..., p-R_m(pdb)))$\rangle$ true

Like in the case of *put* we call the left side. The error branches can be closed. We call the right side. All *legal-...#* calls can be eliminated once more until those which deal with relations containing the entity type E_i. For those we use the following lemmas to finish the proof:

1.
$\langle$legal-R_j#(rel_j, $sent_i$, $sent_{r_{j2}}$; b)$\rangle$ b = tt,
$\langle$in-fst-R_j#(key-E_i(ent_i), rel_j; b)$\rangle$ b = ff,
ent_i $\neq$ error-E_i
$\vdash$
$\langle$legal-R_j#(rel_j, $sent_i$ $-_{E_i}$ ent_i, $sent_{r_{j2}}$;b)$\rangle$ b = tt

2.
$\langle$legal-R_k#(rel_k, $sent_{r_{k1}}$, $sent_i$;b)$\rangle$ b = tt,
$\langle$in-snd-R_k#(key-E_i(ent_i), rel_k; b)$\rangle$ b = ff,
ent_i $\neq$ error-E_i
$\vdash$
$\langle$legal-R_k#(rel_k, $sent_{r_{k1}}$, $sent_i$ $+_{E_i}$ ent_i;b)$\rangle$ b = tt

Like above we use Noetherian induction over *rel_j (rel_k)* to prove these lemmas. As a well founded order we use the *realsub* relation of the specification *orderset*. Before starting the induction we make a case distinction whether *ent_i* is in the set *$sent_i$* or not. In the negative case we are finished and in the positive case we begin the inductive proof. In the base case if *rel_j* is empty both calls of *legal-...#* yield *tt*. In the other case we unfold the left side call of *in-#* and then the left side call of *legal-...#*. After these we unfold the right side diamond, all the if-statements of the right side can be decided. Then we can apply the induction hypothesis because not empty rel_j implies $rest\text{-}R_j(rel_j)$ *realsub* rel_j and finish the proof of these lemmas.

7. Termination of *get-*E_i*#*

$\langle$rs#(pdb)$\rangle$ true $\vdash$ $\langle$get-E_i#(key$_i$, pdb;ent$_i$)$\rangle$ true

Proof: This goal can be proved by trivial unfold steps.

8. Termination of *update-*E_i*#*

$\langle$rs#(pdb)$\rangle$ true $\vdash$ $\langle$update-E_i#(key$_i$, ent$_i$, pdb;pdb$_1$)$\rangle$ $\langle$rs#(pdb$_1$)$\rangle$ true

Proof: At first we unfold the first right diamond formula. After closing all the error cases we get the following goal:

$\langle$rs#(pdb)$\rangle$ true,
key-E_i(ent$_i$) = key$_i$, ent$_i$ $\neq$ error-E_i, pdb $\neq$ p-error-db,
ent$i$-$_1$ = sel-E_i(key$_i$, p-ent-E_i(pdb)), ent$i$-$_1$ $\neq$ error-E_i
$\vdash$
$\langle$rs#(p-mk-db(p-ent-E_1(pdb), ...,
p-ent-E_i(pdb) -$_{E_i}$ ent$i$-$_1$ +$_{E_i}$ ent$_i$, ...,
p-ent-E_n(pdb),
p-R_1(pdb), ..., p-R_m(pdb)))$\rangle$ true

Now we call the left side and then the right side. All the *legal-...#* procedures can be eliminated until those containing the entity type E_i. To prove these we use the following lemmas:

1.
$\langle$legal-R_j#(rel$_j$, sent$_i$, sent$_{r_{j2}}$; b)$\rangle$ b = tt,
key-E_i(ent$_i$) = key-E_i(ent$i$-$_1$),
sent$_i$ -$_{E_i}$ ent$i$-$_1$ +$_{E_i}$ ent$_i$ $\neq$ errorset-E_i,
$\vdash$
$\langle$legal-R_j#(rel$_j$, sent$_i$ -$_{E_i}$ ent$i$-$_1$ +$_{E_i}$ ent$_i$, sent$_{r_{j2}}$;b)$\rangle$ b = tt

2.
$\langle$legal-R_k#(rel$_k$, sent$_{r_{k1}}$, sent$_i$;b)$\rangle$ b = tt,
key-E_i(ent$_i$) = key-E_i(ent$i$-$_1$),
sent$_i$ -$_{E_i}$ ent$i$-$_1$ +$_{E_i}$ ent$_i$ $\neq$ errorset-E_i,
$\vdash$
$\langle$legal-R_k#(rel$_k$, sent$_{r_{k1}}$, sent$_i$ -$_{E_i}$ ent$i$-$_1$ +$_{E_i}$ ent$_i$;b)$\rangle$ b = tt

Like before we prove this goals by induction over rel_j (rel_k).

9. Termination of *est-*R_i*#*

$\langle$rs#(pdb)$\rangle$ true $\vdash$ $\langle$est-R_i#(pdb, entr$_{i1}$-$_1$, entr$_{i2}$-$_2$;pdb$_1$)$\rangle$ $\langle$rs#(pdb$_1$)$\rangle$ true

Proof: Like above we unfold the call of *est-*R_i*#* and then the left side before the right side. To finish the proof we need once more a lemma for the *legal-...#* functions which can be proved by induction too.

$\langle$legal-R_i#(rel$_i$, sent$_{r_{i1}}$, sent$_{r_{i2}}$; b)$\rangle$ b = tt,
sel-ent-$E_{r_{j1}}$(fst-R_i(k$_i$), sent$_{r_{i1}}$) $\neq$ error-$E_{r_{i1}}$,
sel-ent-$E_{r_{j2}}$(snd-R_i(k$_i$), sent$_{r_{i2}}$) $\neq$ error-$E_{r_{i2}}$,
$\vdash$
$\langle$legal-R_i#(rel$_i$ +$_{R_i}$ k$_i$,$_{r_{i1}}$, sent$_{r_{i2}}$;b)$\rangle$ b = tt

10. Termination of *rel-R_i#*

$$\langle \text{rs\#(pdb)} \rangle \text{ true} \vdash \langle \text{rel-R}_i\text{\#(pdb, entr}_{i1}\text{-}_1\text{, entr}_{i2}\text{-}_2\text{;pdb}_1) \rangle \ \langle \text{rs\#(pdb}_1) \rangle \text{ true}$$

Proof: This goal can be proved in the same way as all goals before we only need the following lemma:

$$\langle \text{legal-R}_i\text{\#(rel}_i\text{, sent}_{r_{i1}}\text{, sent}_{r_{i2}}\text{; b)} \rangle \text{ b = tt,}$$
$$\vdash$$
$$\langle \text{legal-R}_i\text{\#(rel}_i \text{ -}_{R_i} \text{ k}_i \text{ ,}_{r_{i1}}\text{, sent}_{r_{i2}}\text{;b)} \rangle \text{ b = tt}$$

To prove this lemma we use induction. Before doing so we distinguish three cases.

(a) k_i is not a member of rel_i. We don't need induction in this case.

(b) k_i is in rel_i and k_i is the minimal element. We don't need induction in this case.

(c) k_i is in rel_i and k_i is not the minimal element. In this case we need induction over rel_i. We do this proof like the other inductive proofs before.

11. Termination of *R_i#*

$$\langle \text{rs\#(pdb)} \rangle \text{ true} \vdash \langle \text{R}_i\text{\#(pdb, entr}_{i1}\text{-}_1\text{, entr}_{i2}\text{-}_2\text{;b)} \rangle \text{ true}$$

Proof: This goal can be proved by trivial unfold steps without additional lemmas.

Proof Obligations Guaranteeing the Right Behavior (iii)

1. Defindedness of *mk-db*

$$\vdash$$

$$
\begin{aligned}
& \neg \ \langle \text{mk-db\#(sent}_1, \ldots, \text{sent}_n, \text{rel}_1, \ldots, \text{rel}_m\text{;pdb}_0) \rangle \\
& \langle \text{error-db\#(;pdb}_2) \rangle \, \text{pdb}_0 = \text{pdb}_2 \\
\leftrightarrow\ & (\forall \text{ ent1-}_1, \text{ent1-}_2.\ \text{ent1-}_1 \text{ in-E}_1 \text{ sent}_1 \\
& \qquad \wedge \text{ent1-}_2 \text{ in-E}_1 \text{ sent}_1 \\
& \qquad \wedge \text{ent1-}_1 \neq \text{ent1-}_2 \\
& \qquad \rightarrow \text{key-E}_1(\text{ent1-}_1) \neq \text{key-E}_1(\text{ent1-}_2)) \\
& \wedge \\
& \vdots \\
& \wedge (\forall \text{ ent}n\text{-}_1, \text{ent}n\text{-}_2.\ \text{ent}n\text{-}_1 \text{ in-E}_n \text{ sent}_n \\
& \qquad \wedge \text{ent}n\text{-}_2 \text{ in-E}_n \text{ sent}_n \\
& \qquad \wedge \text{ent}n\text{-}_1 \neq \text{ent}n\text{-}_2 \\
& \qquad \rightarrow \text{key-E}_n(\text{ent}n\text{-}_1) \neq \text{key-E}_n(\text{ent}n\text{-}_2)) \\
& \wedge (\forall \text{ k1-}_1, \text{k1-}_2.\text{mk-R}_1(\text{k1-}_1, \text{k1-}_2) \text{ in-R}_1 \text{ rel}_1 \\
& \qquad \rightarrow (\exists \text{ entr}_{11}\text{-}_1, \text{entr}_{12}\text{-}_2.\text{entr}_{11}\text{-}_1 \text{ in-E}_{r_{11}} \text{ sent}_{r_{11}} \\
& \qquad\qquad \wedge \text{entr}_{12}\text{-}_2 \text{ in-E}_{r_{12}} \text{ sent}_{r_{12}} \\
& \qquad\qquad \wedge \text{key-E}_{r_{11}}(\text{entr}_{11}\text{-}_1) = \text{k}r_{m2}\text{-}_1 \\
& \qquad\qquad \wedge \text{key-E}_{r_{12}}(\text{entr}_{12}\text{-}_2) = \text{k}r_{m2}\text{-}_2)) \\
& \vdots \\
& \wedge (\forall \text{ k}m\text{-}_1, \text{k}m\text{-}_2.\ \text{mk-R}_m(\text{k}m\text{-}_1, \text{k}m\text{-}_2) \text{ in-R}_m \text{ rel}_m \\
& \qquad \rightarrow (\exists \text{ entr}_{m1}\text{-}_1, \text{entr}_{m2}\text{-}_2.\text{entr}_{m1}\text{-}_1 \text{ in-E}_{r_{m1}} \text{ sent}_{r_{m1}} \\
& \qquad\qquad \wedge \text{entr}_{m2}\text{-}_2 \text{ in-E}_{r_{m2}} \text{ sent}_{r_{m2}} \\
& \qquad\qquad \wedge \text{key-E}_{r_{m1}}(\text{entr}_{m1}\text{-}_1) = \text{k}m\text{-}_1 \\
& \qquad\qquad \wedge \text{key-E}_{r_{m2}}(\text{entr}_{m2}\text{-}_2) = \text{k}m\text{-}_2)) \\
& \wedge \text{sent}_1 \neq \text{errorset-E}_1 \\
& \vdots \\
& \wedge \text{sent}_n \neq \text{errorset-E}_n
\end{aligned}
$$

Proof: To prove this goal we shift the succedent to the antecedent and eliminate the equivalence by two implications. Then we make a case distinction on the implication with the diamonds on the right hand side. We got two new goals:

1.

$$
\begin{array}{l}
\langle \text{mk-db\#}(\text{sent}_1, \ldots, \text{sent}_n, \text{rel}_1, \ldots, \text{rel}_m;\text{pdb}_0)\rangle \\
\langle \text{error-db\#}(;\text{pdb}_2)\rangle\, \text{pdb}_0 = \text{pdb}_2 \\
\rightarrow (\forall\ \text{ent1-}_1, \text{ent1-}_2.\ \text{ent1-}_1 \text{ in-E}_1 \text{ sent}_1 \\
\qquad \wedge \text{ent1-}_2 \text{ in-E}_1 \text{ sent}_1 \\
\qquad \wedge \text{ent1-}_1 \neq \text{ent1-}_2 \\
\qquad \rightarrow \text{key-E}_1(\text{ent1-}_1) \neq \text{key-E}_1(\text{ent1-}_2)) \\
\wedge \\
\vdots \\
\wedge (\forall\ \text{ent}n\text{-}_1, \text{ent}n\text{-}_2.\ \text{ent}n\text{-}_1 \text{ in-E}_n \text{ sent}_n \\
\qquad \wedge \text{ent}n\text{-}_2 \text{ in-E}_n \text{ sent}_n \\
\qquad \wedge \text{ent}n\text{-}_1 \neq \text{ent}n\text{-}_2 \\
\qquad \rightarrow \text{key-E}_n(\text{ent}n\text{-}_1) \neq \text{key-E}_n(\text{ent}n\text{-}_2)) \\
\wedge (\forall\ \text{k1-}_1, \text{k1-}_2.\ \text{mk-R}_1(\text{k1-}_1, \text{k1-}_2) \text{ in-R}_1 \text{ rel}_1 \\
\qquad \rightarrow (\exists\ \text{ent}r_{11}\text{-}_1, \text{ent}r_{12}\text{-}_2.\text{ent}r_{11}\text{-}_1 \text{ in-E}_{r_{11}} \text{ sent}_{r_{11}} \\
\qquad\qquad \wedge \text{ent}r_{12}\text{-}_2 \text{ in-E}_{r_{12}} \text{ sent}_{r_{12}} \\
\qquad\qquad \wedge \text{key-E}_{r_{11}}(\text{ent}r_{11}\text{-}_1) = \text{k}r_{m2}\text{-}_1 \\
\qquad\qquad \wedge \text{key-E}_{r_{12}}(\text{ent}r_{12}\text{-}_2) = \text{k}r_{m2}\text{-}_2)) \\
\wedge \\
\vdots \\
\wedge (\forall\ \text{k}m\text{-}_1, \text{k}m\text{-}_2.\ \text{mk-R}_m(\text{k}m\text{-}_1, \text{k}m\text{-}_2) \text{ in-R}_m \text{ rel}_m \\
\qquad \rightarrow (\exists\ \text{ent}r_{m1}\text{-}_1, \text{ent}r_{m2}\text{-}_2.\text{ent}r_{m1}\text{-}_1 \text{ in-E}_{r_{m1}} \text{ sent}_{r_{m1}} \\
\qquad\qquad \wedge \text{ent}r_{m2}\text{-}_2 \text{ in-E}_{r_{m2}} \text{ sent}_{r_{m2}} \\
\qquad\qquad \wedge \text{key-E}_{r_{m1}}(\text{ent}r_{m1}\text{-}_1) = \text{k}m\text{-}_1 \\
\qquad\qquad \wedge \text{key-E}_{r_{m2}}(\text{ent}r_{m2}\text{-}_2) = \text{k}m\text{-}_2)) \\
\wedge \text{sent}_1 \neq \text{errorset-E}_1 \\
\vdots \\
\wedge \text{sent}_n \neq \text{errorset-E}_n, \\
\neg ((\forall\ \text{ent1-}_1, \text{ent1-}_2.\ \text{ent1-}_1 \text{ in-E}_1 \text{ sent}_1 \\
\qquad \wedge \text{ent1-}_2 \text{ in-E}_1 \text{ sent}_1 \\
\qquad \wedge \text{ent1-}_1 \neq \text{ent1-}_2 \\
\qquad \rightarrow \text{key-E}_1(\text{ent1-}_1) \neq \text{key-E}_1(\text{ent1-}_2)) \\
\wedge \\
\vdots \\
\wedge (\forall\ \text{ent}n\text{-}_1, \text{ent}n\text{-}_2.\ \text{ent}n\text{-}_1 \text{ in-E}_n \text{ sent}_n \\
\qquad \wedge \text{ent}n\text{-}_2 \text{ in-E}_n \text{ sent}_n \\
\qquad \wedge \text{ent}n\text{-}_1 \neq \text{ent}n\text{-}_2 \\
\qquad \rightarrow \text{key-E}_n(\text{ent}n\text{-}_1) \neq \text{key-E}_n(\text{ent}n\text{-}_2)) \\
\wedge (\forall\ \text{k1-}_1, \text{k1-}_2.\ \text{mk-R}_1(\text{k1-}_1, \text{k1-}_2) \text{ in-R}_1 \text{ rel}_1 \\
\qquad \rightarrow (\exists\ \text{ent}r_{11}\text{-}_1, \text{ent}r_{12}\text{-}_2.\text{ent}r_{11}\text{-}_1 \text{ in-E}_{r_{11}} \text{ sent}_{r_{11}} \\
\qquad\qquad \wedge \text{ent}r_{12}\text{-}_2 \text{ in-E}_{r_{12}} \text{ sent}_{r_{12}} \\
\qquad\qquad \wedge \text{key-E}_{r_{11}}(\text{ent}r_{11}\text{-}_1) = \text{k}r_{m2}\text{-}_1 \\
\qquad\qquad \wedge \text{key-E}_{r_{12}}(\text{ent}r_{12}\text{-}_2) = \text{k}r_{m2}\text{-}_2)) \\
\wedge \\
\vdots \\
\wedge (\forall\ \text{k}m\text{-}_1, \text{k}m\text{-}_2.\ \text{mk-R}_m(\text{k}m\text{-}_1, \text{k}m\text{-}_2) \text{ in-R}_m \text{ rel}_m \\
\qquad \rightarrow (\exists\ \text{ent}r_{m1}\text{-}_1, \text{ent}r_{m2}\text{-}_2.\text{ent}r_{m1}\text{-}_1 \text{ in-E}_{r_{m1}} \text{ sent}_{r_{m1}} \\
\qquad\qquad \wedge \text{ent}r_{m2}\text{-}_2 \text{ in-E}_{r_{m2}} \text{ sent}_{r_{m2}} \\
\qquad\qquad \wedge \text{key-E}_{r_{m1}}(\text{ent}r_{m1}\text{-}_1) = \text{k}m\text{-}_1
\end{array}
$$

$\wedge\ \text{key-E}_{r_{m2}}(\text{ent}r_{m2}\text{-}_2) = \text{k}m\text{-}_2))$

$\wedge\ \text{sent}_1 \neq \text{errorset-E}_1$

$\vdots$

$\wedge\ \text{sent}_n \neq \text{errorset-E}_n\ )$

$\vdash$

2.

$\langle \text{mk-db\#}(\text{sent}_1, \ldots, \text{sent}_n, \text{rel}_1, \ldots, \text{rel}_m; \text{pdb}_0)\rangle$

$\langle \text{error-db\#}(;\text{pdb}_2)\rangle\, \text{pdb}_0 = \text{pdb}_2$

$\rightarrow (\forall\ \text{ent1-}_1, \text{ent1-}_2.\ \text{ent1-}_1 \text{ in-E}_1 \text{ sent}_1$

$\wedge\ \text{ent1-}_2 \text{ in-E}_1 \text{ sent}_1$

$\wedge\ \text{ent1-}_1 \neq \text{ent1-}_2$

$\rightarrow \text{key-E}_1(\text{ent1-}_1) \neq \text{key-E}_1(\text{ent1-}_2))$

$\wedge$

$\vdots$

$\wedge\ (\forall\ \text{ent}n\text{-}_1, \text{ent}n\text{-}_2.\ \text{ent}n\text{-}_1 \text{ in-E}_n \text{ sent}_n$

$\wedge\ \text{ent}n\text{-}_2 \text{ in-E}_n \text{ sent}_n$

$\wedge\ \text{ent}n\text{-}_1 \neq \text{ent}n\text{-}_2$

$\rightarrow \text{key-E}_n(\text{ent}n\text{-}_1) \neq \text{key-E}_n(\text{ent}n\text{-}_2))$

$\wedge\ (\forall\ \text{k1-}_1, \text{k1-}_2.\ \text{mk-R}_1(\text{k1-}_1, \text{k1-}_2) \text{ in-R}_1 \text{ rel}_1$

$\rightarrow (\exists\ \text{ent}r_{11}\text{-}_1, \text{ent}r_{12}\text{-}_2.\text{ent}r_{11}\text{-}_1 \text{ in-E}_{r_{11}} \text{ sent}_{r_{11}}$

$\wedge\ \text{ent}r_{12}\text{-}_2 \text{ in-E}_{r_{12}} \text{ sent}_{r_{12}}$

$\wedge\ \text{key-E}_{r_{11}}(\text{ent}r_{11}\text{-}_1) = \text{k}r_{m2}\text{-}_1$

$\wedge\ \text{key-E}_{r_{12}}(\text{ent}r_{12}\text{-}_2) = \text{k}r_{m2}\text{-}_2))$

$\wedge$

$\vdots$

$\wedge\ (\forall\ \text{k}m\text{-}_1, \text{k}m\text{-}_2.\ \text{mk-R}_m(\text{k}m\text{-}_1, \text{k}m\text{-}_2) \text{ in-R}_m \text{ rel}_m$

$\rightarrow (\exists\ \text{ent}r_{m1}\text{-}_1, \text{ent}r_{m2}\text{-}_2.\text{ent}r_{m1}\text{-}_1 \text{ in-E}_{r_{m1}} \text{ sent}_{r_{m1}}$

$\wedge\ \text{ent}r_{m2}\text{-}_2 \text{ in-E}_{r_{m2}} \text{ sent}_{r_{m2}}$

$\wedge\ \text{key-E}_{r_{m1}}(\text{ent}r_{m1}\text{-}_1) = \text{k}m\text{-}_1$

$\wedge\ \text{key-E}_{r_{m2}}(\text{ent}r_{m2}\text{-}_2) = \text{k}m\text{-}_2))$

$\wedge\ \text{sent}_1 \neq \text{errorset-E}_1$

$\vdots$

$\wedge\ \text{sent}_n \neq \text{errorset-E}_n,$

$\langle \text{mk-db\#}(\text{sent}_1, \ldots, \text{sent}_n, \text{rel}_1, \ldots, \text{rel}_m; \text{pdb}_0)\rangle$

$\langle \text{error-db\#}(;\text{pdb}_2)\rangle\, \text{pdb}_0 = \text{pdb}_2$

$\vdash$

Working on the first goal, we make one more case distinction on the remaining implication. So the following goal remains:

$\neg\ ((\forall\ \text{ent1-}_1, \text{ent1-}_2.\ \text{ent1-}_1 \text{ in-E}_1 \text{ sent}_1$

$\wedge\ \text{ent1-}_2 \text{ in-E}_1 \text{ sent}_1$

$\wedge\ \text{ent1-}_1 \neq \text{ent1-}_2$

$\rightarrow \text{key-E}_1(\text{ent1-}_1) \neq \text{key-E}_1(\text{ent1-}_2))$

$\wedge$

$\vdots$

$\wedge\ (\forall\ \text{ent}n\text{-}_1, \text{ent}n\text{-}_2.\ \text{ent}n\text{-}_1 \text{ in-E}_n \text{ sent}_n$

$\wedge\ \text{ent}n\text{-}_2 \text{ in-E}_n \text{ sent}_n$

$\wedge\ \text{ent}n\text{-}_1 \neq \text{ent}n\text{-}_2$

$\rightarrow \text{key-E}_n(\text{ent}n\text{-}_1) \neq \text{key-E}_n(\text{ent}n\text{-}_2))$

$\wedge\ (\forall\ \text{k1-}_1, \text{k1-}_2.\ \text{mk-R}_1(\text{k1-}_1, \text{k1-}_2) \text{ in-R}_1 \text{ rel}_1$

$\rightarrow (\exists\ \text{ent}r_{11}\text{-}_1, \text{ent}r_{12}\text{-}_2.\text{ent}r_{11}\text{-}_1 \text{ in-E}_{r_{11}} \text{ sent}_{r_{11}}$

$$\begin{aligned}
&\qquad\qquad\qquad\qquad \wedge\ \mathrm{ent}r_{12}\text{-}_2\ \mathrm{in\text{-}E}_{r_{12}}\ \mathrm{sent}_{r_{12}}\\
&\qquad\qquad\qquad\qquad \wedge\ \mathrm{key\text{-}E}_{r_{11}}(\mathrm{ent}r_{11}\text{-}_1) = \mathrm{k}r_{m2}\text{-}_1\\
&\qquad\qquad\qquad\qquad \wedge\ \mathrm{key\text{-}E}_{r_{12}}(\mathrm{ent}r_{12}\text{-}_2) = \mathrm{k}r_{m2}\text{-}_2))\\
&\wedge\\
&\vdots\\
&\wedge\ (\forall\ \mathrm{k}m\text{-}_1,\ \mathrm{k}m\text{-}_2.\ \mathrm{mk\text{-}R}_m(\mathrm{k}m\text{-}_1,\ \mathrm{k}m\text{-}_2)\ \mathrm{in\text{-}R}_m\ \mathrm{rel}_m\\
&\qquad \rightarrow (\exists\ \mathrm{ent}r_{m1}\text{-}_1,\ \mathrm{ent}r_{m2}\text{-}_2.\mathrm{ent}r_{m1}\text{-}_1\ \mathrm{in\text{-}E}_{r_{m1}}\ \mathrm{sent}_{r_{m1}}\\
&\qquad\qquad\qquad \wedge\ \mathrm{ent}r_{m2}\text{-}_2\ \mathrm{in\text{-}E}_{r_{m2}}\ \mathrm{sent}_{r_{m2}}\\
&\qquad\qquad\qquad \wedge\ \mathrm{key\text{-}E}_{r_{m1}}(\mathrm{ent}r_{m1}\text{-}_1) = \mathrm{k}m\text{-}_1\\
&\qquad\qquad\qquad \wedge\ \mathrm{key\text{-}E}_{r_{m2}}(\mathrm{ent}r_{m2}\text{-}_2) = \mathrm{k}m\text{-}_2))\\
&\wedge\ \mathrm{sent}_1 \neq \mathrm{errorset\text{-}E}_1\\
&\vdots\\
&\wedge\ \mathrm{sent}_n \neq \mathrm{errorset\text{-}E}_n\ )\\
\vdash&\\
&\langle \mathrm{mk\text{-}db\#}(\mathrm{sent}_1, \ldots, \mathrm{sent}_n, \mathrm{rel}_1, \ldots, \mathrm{rel}_m;\mathrm{pdb}_0)\rangle\\
&\langle \mathrm{error\text{-}db\#}(;\mathrm{pdb}_2)\rangle\, \mathrm{pdb}_0 = \mathrm{pdb}_2
\end{aligned}$$

We now unfold the succedent. During the unfold steps we shift the *legal-R_i#* calls to the left side using the lemma about termination. The result is the following goal:

$$\begin{aligned}
&\langle \mathrm{legal\text{-}R_1\#}(\mathrm{rel}_1,\ \mathrm{sent}_{r_{11}},\ \mathrm{sent}_{r_{12}};\ \mathrm{b}_0)\rangle\, \mathrm{b}_0 = \mathrm{tt},\\
&\vdots\\
&\langle \mathrm{legal\text{-}R_m\#}(\mathrm{rel}_m,\ \mathrm{sent}_{r_{m1}},\ \mathrm{sent}_{r_{m2}};\ \mathrm{b}_0)\rangle\, \mathrm{b}_0 = \mathrm{tt},\\
&(\exists\ \mathrm{ent1}\text{-}_1,\ \mathrm{ent1}\text{-}_2.\ \mathrm{ent1}\text{-}_1\ \mathrm{in\text{-}E}_1\ \mathrm{sent}_1\\
&\qquad\qquad \wedge\ \mathrm{ent1}\text{-}_2\ \mathrm{in\text{-}E}_1\ \mathrm{sent}_1\\
&\qquad\qquad \wedge\ \mathrm{ent1}\text{-}_1 \neq \mathrm{ent1}\text{-}_2\\
&\qquad\qquad \wedge\ \mathrm{key\text{-}E}_1(\mathrm{ent1}\text{-}_1) = \mathrm{key\text{-}E}_1(\mathrm{ent1}\text{-}_2))\\
&\vee\\
&\vdots\\
&\vee\ (\exists\ \mathrm{ent}n\text{-}_1,\ \mathrm{ent}n\text{-}_2.\ \mathrm{ent}n\text{-}_1\ \mathrm{in\text{-}E}_n\ \mathrm{sent}_n\\
&\qquad\qquad \wedge\ \mathrm{ent}n\text{-}_2\ \mathrm{in\text{-}E}_n\ \mathrm{sent}_n\\
&\qquad\qquad \wedge\ \mathrm{ent}n\text{-}_1 \neq \mathrm{ent}n\text{-}_2\\
&\qquad\qquad \wedge\ \mathrm{key\text{-}E}_n(\mathrm{ent}n\text{-}_1) = \mathrm{key\text{-}E}_n(\mathrm{ent}n\text{-}_2))\\
&\vee\ (\exists\ \mathrm{k1}\text{-}_1,\ \mathrm{k1}\text{-}_2.\ \mathrm{mk\text{-}R}_1(\mathrm{k1}\text{-}_1,\ \mathrm{k1}\text{-}_2)\ \mathrm{in\text{-}R}_1\ \mathrm{rel}_1\\
&\qquad\qquad \wedge\ (\forall\ \mathrm{ent}r_{11}\text{-}_1,\ \mathrm{ent}r_{12}\text{-}_2.\neg(\mathrm{ent}r_{11}\text{-}_1\ \mathrm{in\text{-}E}_{r_{11}}\ \mathrm{sent}_{r_{11}}\\
&\qquad\qquad\qquad\qquad \wedge\ \mathrm{ent}r_{12}\text{-}_2\ \mathrm{in\text{-}E}_{r_{12}}\ \mathrm{sent}_{r_{12}}\\
&\qquad\qquad\qquad\qquad \wedge\ \mathrm{key\text{-}E}_{r_{11}}(\mathrm{ent}r_{11}\text{-}_1) = \mathrm{k}r_{m2}\text{-}_1\\
&\qquad\qquad\qquad\qquad \wedge\ \mathrm{key\text{-}E}_{r_{12}}(\mathrm{ent}r_{12}\text{-}_2) = \mathrm{k}r_{m2}\text{-}_2))\\
&\vee\\
&\vdots\\
&\vee\ (\exists\ \mathrm{k}m\text{-}_1,\ \mathrm{k}m\text{-}_2.\ \mathrm{mk\text{-}R}_m(\mathrm{k}m\text{-}_1,\ \mathrm{k}m\text{-}_2)\ \mathrm{in\text{-}R}_m\ \mathrm{rel}_m\\
&\qquad\qquad \wedge\ (\forall\ \mathrm{ent}r_{m1}\text{-}_1,\ \mathrm{ent}r_{m2}\text{-}_2.\neg\mathrm{ent}r_{m1}\text{-}_1\ \mathrm{in\text{-}E}_{r_{m1}}\ \mathrm{sent}_{r_{m1}}\\
&\qquad\qquad\qquad\qquad \wedge\ \mathrm{ent}r_{m2}\text{-}_2\ \mathrm{in\text{-}E}_{r_{m2}}\ \mathrm{sent}_{r_{m2}}\\
&\qquad\qquad\qquad\qquad \wedge\ \mathrm{key\text{-}E}_{r_{m1}}(\mathrm{ent}r_{m1}\text{-}_1) = \mathrm{k}m\text{-}_1\\
&\qquad\qquad\qquad\qquad \wedge\ \mathrm{key\text{-}E}_{r_{m2}}(\mathrm{ent}r_{m2}\text{-}_2) = \mathrm{k}m\text{-}_2)),\\
&\mathrm{sent}_1 \neq \mathrm{errorset\text{-}E}_1,\ \ldots,\ \mathrm{sent}_n \neq \mathrm{errorset\text{-}E}_n\\
\vdash&
\end{aligned}$$

To close this goal we make a case distinction and using the following lemmas:

$$\begin{aligned}
&\mathrm{mk\text{-}R}_i(\mathrm{k}i\text{-}_1,\ \mathrm{k}i\text{-}_2)\ \mathrm{in\text{-}R}_i\ \mathrm{rel}_i\\
&\wedge\ (\forall\ \mathrm{ent}r_{i1}\text{-}_1,\ \mathrm{ent}r_{i2}\text{-}_2.\neg\mathrm{ent}r_{m1}\text{-}_1\ \mathrm{in\text{-}E}_{r_{m1}}\ \mathrm{sent}_{r_{m1}}\\
&\qquad\qquad \wedge\ \mathrm{ent}r_{m2}\text{-}_2\ \mathrm{in\text{-}E}_{r_{m2}}\ \mathrm{sent}_{r_{m2}}\\
&\qquad\qquad \wedge\ \mathrm{key\text{-}E}_{r_{m1}}(\mathrm{ent}r_{m1}\text{-}_1) = \mathrm{k}m\text{-}_1
\end{aligned}$$

$$\wedge \text{ key-E}_{r_{m2}}(\text{entr}_{m2\text{-}2}) = \text{k}m_{\text{-}2}))),$$

$\vdash$

$\langle \text{legal-R}_i\#(\text{rel}_i, \text{sent}_{r_{i1}}, \text{sent}_{r_{i2}}; \text{b}_0)\rangle\, \text{b}_0 = \text{ff}$

To prove these lemmas we use induction over *rel_i* like above.

Or we use the fact that the following formulas are equivalent to false according to the specification *coded-set*.

$$\begin{aligned}
&\text{ent}i_{\text{-}1} \text{ in-E}_i \text{ sent}_i\\
&\wedge \text{ ent}i_{\text{-}2} \text{ in-E}_i \text{ sent}_i\\
&\wedge \text{ ent}i_{\text{-}1} \neq \text{ent}i_{\text{-}2}\\
&\wedge \text{ key-E}_i(\text{ent}i_{\text{-}1}) = \text{key-E}_i(\text{ent}i_{\text{-}2})
\end{aligned}$$

Now we can work on the second goal.

$$\begin{aligned}
&\langle \text{mk-db}\#(\text{sent}_1, \ldots, \text{sent}_n, \text{rel}_1, \ldots, \text{rel}_m; \text{pdb}_0)\rangle\\
&\langle \text{error-db}\#(;\text{pdb}_2)\rangle\, \text{pdb}_0 = \text{pdb}_2\\
\rightarrow\ &(\forall\ \text{ent1}_{\text{-}1}, \text{ent1}_{\text{-}2}.\ \text{ent1}_{\text{-}1} \text{ in-E}_1 \text{ sent}_1\\
&\qquad \wedge \text{ ent1}_{\text{-}2} \text{ in-E}_1 \text{ sent}_1\\
&\qquad \wedge \text{ ent1}_{\text{-}1} \neq \text{ent1}_{\text{-}2}\\
&\qquad \rightarrow \text{key-E}_1(\text{ent1}_{\text{-}1}) \neq \text{key-E}_1(\text{ent1}_{\text{-}2}))\\
&\wedge\\
&\vdots\\
&\wedge\ (\forall\ \text{ent}n_{\text{-}1}, \text{ent}n_{\text{-}2}.\ \text{ent}n_{\text{-}1} \text{ in-E}_n \text{ sent}_n\\
&\qquad \wedge \text{ ent}n_{\text{-}2} \text{ in-E}_n \text{ sent}_n\\
&\qquad \wedge \text{ ent}n_{\text{-}1} \neq \text{ent}n_{\text{-}2}\\
&\qquad \rightarrow \text{key-E}_n(\text{ent}n_{\text{-}1}) \neq \text{key-E}_n(\text{ent}n_{\text{-}2}))\\
&\wedge\ (\forall\ \text{k1}_{\text{-}1}, \text{k1}_{\text{-}2}.\ \text{mk-R}_1(\text{k1}_{\text{-}1}, \text{k1}_{\text{-}2}) \text{ in-R}_1 \text{ rel}_1\\
&\qquad \rightarrow (\exists\ \text{entr}_{11\text{-}1}, \text{entr}_{12\text{-}2}.\text{entr}_{11\text{-}1} \text{ in-E}_{r_{11}} \text{ sent}_{r_{11}}\\
&\qquad\qquad \wedge \text{ entr}_{12\text{-}2} \text{ in-E}_{r_{12}} \text{ sent}_{r_{12}}\\
&\qquad\qquad \wedge \text{ key-E}_{r_{11}}(\text{entr}_{11\text{-}1}) = \text{kr}_{m2\text{-}1}\\
&\qquad\qquad \wedge \text{ key-E}_{r_{12}}(\text{entr}_{12\text{-}2}) = \text{kr}_{m2\text{-}2}))\\
&\wedge\\
&\vdots\\
&\wedge\ (\forall\ \text{k}m_{\text{-}1}, \text{k}m_{\text{-}2}.\ \text{mk-R}_m(\text{k}m_{\text{-}1}, \text{k}m_{\text{-}2}) \text{ in-R}_m \text{ rel}_m\\
&\qquad \rightarrow (\exists\ \text{entr}_{m1\text{-}1}, \text{entr}_{m2\text{-}2}.\text{entr}_{m1\text{-}1} \text{ in-E}_{r_{m1}} \text{ sent}_{r_{m1}}\\
&\qquad\qquad \wedge \text{ entr}_{m2\text{-}2} \text{ in-E}_{r_{m2}} \text{ sent}_{r_{m2}}\\
&\qquad\qquad \wedge \text{ key-E}_{r_{m1}}(\text{entr}_{m1\text{-}1}) = \text{k}m_{\text{-}1}\\
&\qquad\qquad \wedge \text{ key-E}_{r_{m2}}(\text{entr}_{m2\text{-}2}) = \text{k}m_{\text{-}2}))\\
&\wedge\ \text{sent}_1 \neq \text{errorset-E}_1\\
&\vdots\\
&\wedge\ \text{sent}_n \neq \text{errorset-E}_n,\\
&\langle \text{mk-db}\#(\text{sent}_1, \ldots, \text{sent}_n, \text{rel}_1, \ldots, \text{rel}_m; \text{pdb}_0)\rangle\\
&\langle \text{error-db}\#(;\text{pdb}_2)\rangle\, \text{pdb}_0 = \text{pdb}_2\\
&\vdash
\end{aligned}$$

At first we eliminate the implication knowing the left side is true. Then we split the diamond and call *error-db#* before *mk-db#*. After unfolding *mk-db#*. We get the following goal:

$$\begin{aligned}
&\langle \text{legal-R}_1\#(\text{rel}_1, \text{sent}_{r_{11}}, \text{sent}_{r_{12}}; \text{b}_0)\rangle\, \text{b}_0 = \text{ff},\\
&\vdots\\
&\langle \text{legal-R}_m\#(\text{rel}_m, \text{sent}_{r_{m1}}, \text{sent}_{r_{m2}}; \text{b}_0)\rangle\, \text{b}_0 = \text{ff},\\
&\forall\ \text{ent1}_{\text{-}1}, \text{ent1}_{\text{-}2}.\ \text{ent1}_{\text{-}1} \text{ in-E}_1 \text{ sent}_1\\
&\qquad \wedge \text{ ent1}_{\text{-}2} \text{ in-E}_1 \text{ sent}_1
\end{aligned}$$

$\wedge$ ent1-$_1$ $\neq$ ent1-$_2$
$\rightarrow$ key-E$_1$(ent1-$_1$) $\neq$ key-E$_1$(ent1-$_2$),

$\vdots$

$\forall$ ent$n$-$_1$, ent$n$-$_2$. ent$n$-$_1$ in-E$_n$ sent$_n$
$\wedge$ ent$n$-$_2$ in-E$_n$ sent$_n$
$\wedge$ ent$n$-$_1$ $\neq$ ent$n$-$_2$
$\rightarrow$ key-E$_n$(ent$n$-$_1$) $\neq$ key-E$_n$(ent$n$-$_2$),

mk-R$_1$(k1-$_1$, k1-$_2$) in-R$_1$ rel$_1$
$\rightarrow$ ($\exists$ entr_{11}-$_1$, entr_{12}-$_2$.entr_{11}-$_1$ in-E$_{r_{11}}$ sent$_{r_{11}}$
$\wedge$ entr_{12}-$_2$ in-E$_{r_{12}}$ sent$_{r_{12}}$
$\wedge$ key-E$_{r_{11}}$(entr_{11}-$_1$) = kr_{m2}-$_1$
$\wedge$ key-E$_{r_{12}}$(entr_{12}-$_2$) = kr_{m2}-$_2$),

$\vdots$

$\forall$ km-$_1$, km-$_2$. mk-R$_m$(km-$_1$, km-$_2$) in-R$_m$ rel$_m$
$\rightarrow$ ($\exists$ entr_{m1}-$_1$, entr_{m2}-$_2$.entr_{m1}-$_1$ in-E$_{r_{m1}}$ sent$_{r_{m1}}$
$\wedge$ entr_{m2}-$_2$ in-E$_{r_{m2}}$ sent$_{r_{m2}}$
$\wedge$ key-E$_{r_{m1}}$(entr_{m1}-$_1$) = km-$_1$
$\wedge$ key-E$_{r_{m2}}$(entr_{m2}-$_2$) = km-$_2$),

sent$_1$ $\neq$ errorset-E$_1$,

$\vdots$

sent$_n$ $\neq$ errorset-E$_n$,

$\vdash$

To close this goal we use the following lemmas:

$\langle$legal-R$_i$#(rel$_i$, sent$_{r_{i1}}$, sent$_{r_{i2}}$; b$_0$)$\rangle$ b$_0$ = ff

$\vdash$

($\exists$ ki-$_1$, ki-$_2$. mk-R$_i$(ki-$_1$, ki-$_2$) in-R$_i$ rel$_i$
$\wedge$ ($\forall$ entr_{i1}-$_1$, entr_{i2}-$_2$.$\neg$entr_{i1}-$_1$ in-E$_{r_{i1}}$ sent$_{r_{i1}}$
$\wedge$ entr_{i2}-$_2$ in-E$_{r_{i2}}$ sent$_{r_{i2}}$
$\wedge$ key-E$_{r_{i1}}$(entr_{i1}-$_1$) = ki-$_1$
$\wedge$ key-E$_{r_{i2}}$(entr_{i2}-$_2$) = ki-$_2$)))

We prove these lemmas by induction too.

2. Uniqueness of databases.

$\vdash$

$\neg$ $\langle$mk-db#(sent$_1$, ..., sent$_n$, rel$_1$, ..., rel$_m$;pdb$_0$)$\rangle$
$\langle$error-db#(;pdb$_2$)$\rangle$ pdb$_0$ = pdb$_2$
$\rightarrow$ ($\langle$mk-db#(sent$_1$, ..., sent$_n$,
rel$_1$, ..., rel$_m$;pdb$_0$)$\rangle$
$\langle$mk-db#(sent1-$_1$, ..., sent$n$-$_1$,
rel1-$_1$, ..., rel$m$-$_1$;pdb$_2$)$\rangle$
pdb$_0$ = pdb$_2$
$\rightarrow$ sent$_1$ = sent1-$_1$
$\wedge$

$\vdots$

$\wedge$ sent$_n$ = sent$n$-$_1$
$\wedge$ rel$_1$ = rel1-$_1$
$\wedge$

$\vdots$

$\wedge$ rel$_m$ = rel$m$-$_1$)

Proof: At first we normalize the goal.

$\langle$mk-db#(sent$_1$, ..., sent$_n$, rel$_1$, ..., rel$_m$;pdb$_0$)$\rangle$
$\langle$mk-db#(sent1-$_1$, ..., sent$n$-$_1$, rel1-$_1$, ..., rel$m$-$_1$;pdb$_2$)$\rangle$ pdb$_0$ = pdb$_2$,
$\neg$(sent$_1$ = sent1-$_1$ $\wedge$ ... $\wedge$ sent$_n$ = sent$n$-$_1$
$\wedge$ rel$_1$ = rel1-$_1$ $\wedge$ ... $\wedge$ rel$_m$ = rel$m$-$_1$)
$\vdash$
$\langle$mk-db#(sent$_1$, ..., sent$_n$, rel$_1$, ..., rel$_m$;pdb$_0$)$\rangle$ $\langle$error-db#(;pdb$_2$)$\rangle$ pdb$_0$ = pdb$_2$

Then we eliminate the first diamond on the right side by introducing a new variable. Then we call the *error-db#* procedure. This yields the following goal:

$\langle$mk-db#(sent1-$_1$, ..., sent$n$-$_1$, rel1-$_1$, ..., rel$m$-$_1$;pdb$_2$)$\rangle$ pdb$_3$ = pdb$_2$,
$\langle$mk-db#(sent$_1$, ..., sent$_n$, rel$_1$, ..., rel$_m$;pdb$_0$)$\rangle$ pdb$_0$ = pdb$_3$,
$\neg$(sent$_1$ = sent1-$_1$ $\wedge$ ... $\wedge$ sent$_n$ = sent$n$-$_1$
$\wedge$ rel$_1$ = rel1-$_1$ $\wedge$ ... $\wedge$ rel$_m$ = rel$m$-$_1$),
pdb$_3$ $\neq$ p-error-db
$\vdash$

Then we unfold the left side diamonds. Using the fact that $pdb_3 \neq$ *p-error-db* we get

p-mk-db(sent1-$_1$, ..., sent$n$-$_1$, rel1-$_1$, ..., rel$m$-$_1$) = pdb$_3$,
p-mk-db((sent$_1$, ..., sent$_n$, rel$_1$, ..., rel$_m$) = pdb$_3$,
$\neg$(sent$_1$ = sent1-$_1$ $\wedge$ ... $\wedge$ sent$_n$ = sent$n$-$_1$
$\wedge$ rel$_1$ = rel1-$_1$ $\wedge$ ... $\wedge$ rel$_m$ = rel$m$-$_1$)
$\vdash$

and close the goal.

3. The empty database.

$\vdash$

$\langle$empty-db#(;pdb$_0$)$\rangle$
$\langle$mk-db#(emptyset-E$_1$, ...,
emptyset-E$_n$,
emptyrel-R$_1$,...,
emptyrel-R$_m$;pdb$_2$)$\rangle$ pdb$_0$ = pdb$_2$

Proof: This goal can be proved by trivial unfold steps without additional lemmas.

4. The behavior of *ent-E#*, *put-E#* and *del-E#* on defined databases.

$\langle$rs#(pdb)$\rangle$ true
$\vdash$
$\neg$ $\langle$error-db#(;pdb$_0$)$\rangle$ pdb = pdb$_0$
$\wedge$ $\langle$mk-db#(sent$_1$, ..., sent$_n$,
rel$_1$, ..., rel$_m$;pdb$_0$)$\rangle$ pdb = pdb$_0$
$\rightarrow$ $\langle$ent-E$_1$#(pdb;sent1-$_0$)$\rangle$ sent1-$_0$ = sent$_1$
$\wedge$
$\vdots$
$\wedge$ $\langle$ent-E$_n$#(pdb;sent$n$-$_0$)$\rangle$ sent$n$-$_0$ = sent$_n$
$\wedge$ $\langle$put-E$_1$#(ent$_1$, pdb;pdb$_0$)$\rangle$
$\langle$mk-db#(sent$_1$ $+_{E_1}$ ent$_1$, ..., sent$_n$,
rel$_1$, ..., rel$_m$;pdb$_2$)$\rangle$ pdb$_0$ = pdb$_2$

$\wedge$

$\vdots$

$\wedge$ $\langle$put-E$_n$#(ent$_n$, pdb;pdb$_0$)$\rangle$
$\langle$mk-db#(sent$_1$, ..., sent$_n$ $+_{E_n}$ ent$_n$,
rel$_1$, ..., rel$_m$;pdb$_2$)$\rangle$ pdb$_0$ = pdb$_2$
$\wedge$ $\langle$del-E$_1$#(ent$_1$, pdb;pdb$_0$)$\rangle$
$\langle$mk-db#(sent$_1$ $-_{E_1}$ ent$_1$, ..., sent$_n$,
rel$_1$, ..., rel$_m$;pdb$_2$)$\rangle$ pdb$_0$ = pdb$_2$

$\wedge$

$\vdots$

$\wedge$ $\langle$del-E$_n$#(ent$_n$, pdb;pdb$_0$)$\rangle$
$\langle$mk-db#(sent$_1$, ..., sent$_n$ $-_{E_n}$ ent$_n$,
rel$_1$, ..., rel$_m$;pdb$_2$)$\rangle$ pdb$_0$ = pdb$_2$

Proof: To prove this goal we normalize the left side. Then we call the *error-db#* procedure and then the *mk-db#* procedure. So we get the following classes of goals:

1.

$\langle$rs#(p-mk-db(sent$_1$, ..., sent$_n$, rel$_1$, ..., rel$_m$))$\rangle$ true,
$\langle$legal-R$_1$#(rel$_1$, sent$_{r_{11}}$, sent$_{r_{12}}$; b$_0$)$\rangle$ b$_0$ = tt,
$\vdots$
$\langle$legal-R$_m$#(rel$_m$, sent$_{r_{m1}}$, sent$_{r_{m2}}$; b$_0$)$\rangle$ b$_0$ = tt,
sent$_1$ $\neq$ error-E$_1$, ..., sent$_n$ $\neq$ error-E$_n$
$\vdash$
$\langle$ent-E$_i$#(p-mk-db(sent$_1$, ..., sent$_n$, rel$_1$, ..., rel$_m$);sent$i$-$_0$)$\rangle$ sent$i$-$_0$ = sent$_i$

2.

$\langle$rs#(p-mk-db(sent$_1$, ..., sent$_n$, rel$_1$, ..., rel$_m$))$\rangle$ true,
$\langle$legal-R$_1$#(rel$_1$, sent$_{r_{11}}$, sent$_{r_{12}}$; b$_0$)$\rangle$ b$_0$ = tt,
$\vdots$
$\langle$legal-R$_m$#(rel$_m$, sent$_{r_{m1}}$, sent$_{r_{m2}}$; b$_0$)$\rangle$ b$_0$ = tt,
sent$_1$ $\neq$ error-E$_1$, ..., sent$_n$ $\neq$ error-E$_n$
$\vdash$
$\langle$put-E$_i$#(ent$_i$, p-mk-db(sent$_1$, ..., sent$_n$, rel$_1$, ..., rel$_m$);pdb$_0$)$\rangle$
$\langle$mk-db#(sent$_1$, ..., sent$_i$ $+_{E_i}$ ent$_i$, ..., sent$_n$,
rel$_1$, ..., rel$_m$;pdb$_2$)$\rangle$ pdb$_0$ = pdb$_2$

3.

$\langle$rs#(p-mk-db(sent$_1$, ..., sent$_n$, rel$_1$, ..., rel$_m$))$\rangle$ true,
$\langle$legal-R$_1$#(rel$_1$, sent$_{r_{11}}$, sent$_{r_{12}}$; b$_0$)$\rangle$ b$_0$ = tt,
$\vdots$
$\langle$legal-R$_m$#(rel$_m$, sent$_{r_{m1}}$, sent$_{r_{m2}}$; b$_0$)$\rangle$ b$_0$ = tt,
sent$_1$ $\neq$ error-E$_1$, ..., sent$_n$ $\neq$ error-E$_n$
$\vdash$
$\langle$del-E$_i$#(ent$_i$, p-mk-db(sent$_1$, ..., sent$_n$, rel$_1$, ..., rel$_m$);pdb$_0$)$\rangle$
$\langle$mk-db#(sent$_1$,..., sent$_i$ $-_{E_i}$ ent$_i$, ..., sent$_n$,
rel$_1$, ..., rel$_m$;pdb$_2$)$\rangle$ pdb$_0$ = pdb$_2$

To prove the first class we unfold the right side and close the goal.

To prove the second class we unfold the right side too and use the following lemmas, which were used in the termination proofs of *put-E#* too.

1.

$\langle$legal-R_j#(rel_j, $sent_i$, $sent_{r_{j2}}$; b)$\rangle$ b = tt,

$sent_i +_{E_i} ent_i \neq$ errorset-E_i

$\vdash$

$\langle$legal-R_j#(rel_j, $sent_i +_{E_i} ent_i$, $sent_{r_{j2}}$;b)$\rangle$ b = tt

2.

$\langle$legal-R_k#(rel_k, $sent_{r_{k1}}$, $sent_i$;b)$\rangle$ b = tt,

$sent_i +_{E_i} ent_i \neq$ errorset-E_i

$\vdash$

$\langle$legal-R_k#(rel_m, $sent_{r_{k1}}$, $sent_i +_{E_i} ent_i$;b)$\rangle$ b = tt

To prove the third class we unfold the right side once more and shift the calls of the procedures *in-...#* to the left side. Then we split the if-statements. In the positive case the procedure terminates with the result *p-error-db*. We now close this sub goal by using one of the following lemmas and unfolding the procedure *mk-db#*.

1.

$\langle$legal-R_j#(rel_j, $sent_i$, $sent_{r_{j2}}$; b)$\rangle$ b = tt,

$\langle$in-fst-R_j#(key-E_i(ent_i), rel_j; b)$\rangle$ b = tt,

$sent_i -_{E_i} ent_i \neq$ errorset-E_i

$\vdash$

$\langle$legal-R_j#(rel_j, $sent_i -_{E_i} ent_i$, $sent_{r_{j2}}$;b)$\rangle$ b = ff

2.

$\langle$legal-R_k#(rel_j, $sent_{r_{k1}}$, $sent_i$; b)$\rangle$ b = tt,

$\langle$in-snd-R_k#(key-E_i(ent_i), rel_k; b)$\rangle$ b = tt,

$sent_i -_{E_i} ent_i \neq$ errorset-E_i

$\vdash$

$\langle$legal-R_k#(rel_k, $sent_{r_{k1}}$, $sent_i -_{E_i} ent_i$;b)$\rangle$ b = ff

Both lemmas can be proved through induction over rel_j (rel_k).

Returning to the proof of *del*. We are now in the case that every call of *in-...#* yield *ff*. We reach the following goal:

$\langle$rs#(p-mk-db($sent_1$, ..., $sent_n$, rel_1, ..., rel_m))$\rangle$ true,

$\langle$legal-R_1#(rel_1, $sent_{r_{11}}$, $sent_{r_{12}}$; b_0)$\rangle$ b_0 = tt,

$\vdots$

$\langle$legal-R_m#(rel_m, $sent_{r_{m1}}$, $sent_{r_{m2}}$; b_0)$\rangle$ b_0 = tt,

$\vdots$

$\langle$in-fst-R_j#(key-E_i(ent_i),p-R_j(pdb); b_0)$\rangle$ b_0 = ff,

$\vdots$

$\langle$in-snd-R_k#(key-E_i(ent_i), p-R_k(pdb); b_0)$\rangle$ b_0 = ff,

$\vdots$

$sent_1 \neq$ errorset-E_1, ..., $sent_n \neq$ errorset-E_n,

$ent_i \neq$ error-E_1,

$senti\text{-}_1 = sent_i -_{E_i} ent_i$,

$\vdash$

$\langle$del-E$_i$#(ent$_i$, p-mk-db(sent$_1$, ..., sent$_n$, rel$_1$, ..., rel$_m$);pdb$_0$)$\rangle$
$\langle$mk-db#(sent$_1$,..., sent$_i$ -$_{E_i}$ ent$_i$, ..., sent$_n$,
rel$_1$, ..., rel$_m$;pdb$_2$)$\rangle$
p-mk-db(sent$_1$, ..., sent$i$-$_1$, ..., sent$_n$, rel$_1$, ..., rel$_m$) = pdb$_2$

To close this goal and finishing the proof we unfold the right side and use the following lemmas:

$\langle$legal-R$_j$#(rel$_j$, sent$_i$, sent$_{r_{j2}}$; b)$\rangle$ b = tt,
$\langle$in-fst-R$_j$#(key-E$_i$(ent$_i$), rel$_j$; b)$\rangle$ b = ff,
$\langle$in-snd-R$_j$#(key-E$_i$(ent$_i$), rel$_j$; b)$\rangle$ b = ff,
sent$_i$ -$_{E_i}$ ent$_i$ $\neq$ errorset-E$_i$
$\vdash$
$\langle$legal-R$_j$#(rel$_j$, sent$_i$ -$_{E_i}$ ent$_i$, sent$_{r_{j2}}$;b)$\rangle$ b = ff

We prove these lemmas by induction over rel_j once more.

5. The following two goals deals with *get-E#*.

   1.

$\langle$rs#(pdb)$\rangle$ true
$\vdash$
$\neg$ $\langle$get-E$_i$#(key$_i$, pdb;ent$i$-$_0$)$\rangle$ ent$i$-$_0$ = error-E$_i$
$\leftrightarrow$ ($\exists$ ent$_i$. $\langle$ent-E$_i$#(pdb;sent$_i$)$\rangle$ ent$_i$ in-E$_i$ sent$_i$
$\wedge$ key-E$_i$(ent$_i$) = key$_i$)

Proof: At first we normalize the goal and get three sub goals:

1.
$\langle$rs#(pdb)$\rangle$ true
$\vdash$
$\langle$get-E$_i$#(key$_i$, pdb;ent$i$-$_0$)$\rangle$ ent$i$-$_0$ = error-E$_i$,
$\exists$ ent$_i$. $\langle$ent-E$_i$#(pdb;sent$_i$)$\rangle$ ent$_i$ in-E$_i$ sent$_i$ $\wedge$ key-E$_i$(ent$_i$) = key$_i$

2.
$\langle$rs#(pdb)$\rangle$ true,
$\langle$get-E$_i$#(key$_i$, pdb;ent$i$-$_0$)$\rangle$ ent$i$-$_0$ = error-E$_i$,
$\langle$ent-E$_i$#(pdb;sent$_i$)$\rangle$ ent$_i$ in-E$_i$ sent$_i$,
key-E$_i$(ent$i$-$_0$) = key$_i$
$\vdash$

3.
$\langle$rs#(pdb)$\rangle$ true,
$\langle$ent-E$_i$#(pdb;sent$_i$)$\rangle$ ent$i$-$_0$ in-E$_i$ sent$_i$,
key-E$_i$(ent$i$-$_0$) = key$_i$
$\vdash$
$\exists$ ent$_i$. $\langle$ent-E$_i$#(pdb;sent$_i$)$\rangle$ ent$_i$ in-E$_i$ sent$_i$ $\wedge$ key-E$_i$(ent$_i$) = key$_i$

Working on the first goal, we unfold the diamond with *get-*E_i*#* and instantiate the variable ent_i with *sel-*E_i*(*key_i*, p-ent-*E_i*(pdb))* and close this sub goal by unfold steps.
Working on the second goal, we close this goal by trivial unfold steps.
Working on the third goal, we close this goal after instantiating ent_i with *enti-*$_0$. This finishes the proof.

2.

$$\langle \text{rs\#(pdb)} \rangle \text{ true}$$
$$\vdash$$
$$\text{ent}_i \neq \text{error-E}_i$$
$$\rightarrow (\langle \text{get-E}_i\text{\#(key}_i\text{, pdb;ent}i\text{-}_0) \rangle \text{ ent}i\text{-}_0 = \text{ent}_i$$
$$\leftrightarrow \langle \text{ent-E}_i\text{\#(pdb;sent}_i) \rangle \text{ ent}_i \text{ in-E}_i \text{ sent}_i$$
$$\wedge \text{ key-E}_i(\text{ent}_i) = \text{key}_i)$$

Proof: At first we normalize the goal. This yields three sub goals. All the sub goals can be proved by trivial unfold steps.

1.
$\langle \text{rs\#(pdb)} \rangle$ true,
$\langle \text{ent-E}_i\text{\#(pdb;sent}_i) \rangle \text{ ent}_i \text{ in-E}_i \text{ sent}_i$,
$\text{ent}_i \neq \text{error-E}_i$,
$\text{key-E}_i(\text{ent}_i) = \text{key}_i$
$\vdash$
$\langle \text{get-E}_i\text{\#(key}_i\text{, pdb;ent}i\text{-}_0) \rangle \text{ ent}i\text{-}_0 = \text{ent}_i$

2.
$\langle \text{rs\#(pdb)} \rangle$ true,
$\langle \text{get-E}_i\text{\#(key}_i\text{, pdb;ent}i\text{-}_0) \rangle \text{ ent}i\text{-}_0 = \text{ent}_i$,
$\text{ent}_i \neq \text{error-E}_i$
$\vdash$
$\langle \text{ent-E}_i\text{\#(pdb;sent}_i) \rangle \text{ ent}_i \text{ in-E}_i \text{ sent}_i$

3.
$\langle \text{rs\#(pdb)} \rangle$ true,
$\langle \text{get-E}_i\text{\#(key}_i\text{, pdb;ent}i\text{-}_0) \rangle \text{ ent}i\text{-}_0 = \text{ent}_i$,
$\text{ent}_i \neq \text{error-E}_i$,
$\text{key-E}_i(\text{ent}_i) = \text{key}_i$
$\vdash$

6. The following two goals deals with *update-E#*.

1.

$$\langle \text{rs\#(pdb)} \rangle \text{ true}$$
$$\vdash$$
$$\neg \langle \text{update-E}_i\text{\#(key}_i\text{, ent}_i\text{, pdb;pdb}_0) \rangle \langle \text{error-db\#(;pdb}_2) \rangle \text{ pdb}_0 = \text{pdb}_2$$
$$\leftrightarrow (\exists \text{ ent}i\text{-}_2. \langle \text{ent-E}_i\text{\#(pdb;sent}_i) \rangle \text{ ent}i\text{-}_2 \text{ in-E}_i \text{ sent}_i$$
$$\wedge \text{ key-E}_i(\text{ent}i\text{-}_2) = \text{key}_i$$
$$\wedge \text{ key-E}_i(\text{ent}_i) = \text{key}i)$$

Proof: Like above we we first normalize the goal. We get three sub goals:

1.
$\langle \text{rs\#(pdb)} \rangle$ true
$\vdash$
$\exists \text{ ent}i\text{-}_2.\ \langle \text{ent-E}_i\text{\#(pdb;sent}_i) \rangle \text{ ent}i\text{-}_2 \text{ in-E}_i \text{ sent}_i$
$\wedge \text{ key-E}_i(\text{ent}i\text{-}_2) = \text{key}_i \wedge \text{key-E}_i(\text{ent}_i) = \text{key}_i$,
$\langle \text{update-E}_i\text{\#(key}_i\text{, ent}_i\text{, pdb;pdb}_0) \rangle \langle \text{error-db\#(;pdb}_2) \rangle \text{ pdb}_0 = \text{pdb}_2$

2.

$\langle$rs#(pdb)$\rangle$ true,
$\langle$update-E_i#(key$_i$, ent$_i$, pdb;pdb$_0$)$\rangle$ $\langle$error-db#(;pdb$_2$)$\rangle$ pdb$_0$ = pdb$_2$,
$\langle$ent-E_i#(pdb;sent$_i$)$\rangle$ enti_{-1} in-E_i sent$_i$,
key-E_i(enti_{-1}) = key$_i$, key-E_i(ent$_i$) = key$_i$
$\vdash$

3.

$\langle$rs#(pdb)$\rangle$ true,
$\langle$ent-E_i#(pdb;sent$_i$)$\rangle$ enti_{-1} in-E_i sent$_i$,
key-E_i(enti_{-1}) = key$_i$, key-E_i(ent$_i$) = key$_i$
$\vdash$
$\exists$ enti_{-2}. $\langle$ent-E_i#(pdb;sent$_i$)$\rangle$ enti_{-2} in-E_i sent$_i$
$\wedge$ key-E_i(enti_{-2}) = key$_i$ $\wedge$ key-E_i(ent$_i$) = key$_i$

Working on the first goal, we unfold the call of *update-E_i#*. After doing so we instantiate *enti_{-2}* with *sel-E_i(key$_i$, p-ent-E_i(pdb))* and close the goal.

Working on the second goal, we close this goal by repeated unfold steps.

Working on the third goal, instantiating enti_{-2} with enti_{-1} close this goal.

2.

$\langle$rs#(pdb)$\rangle$ true
$\vdash$
$\neg$ $\langle$update-E_i#(key$_i$, ent$_i$, pdb;pdb$_0$)$\rangle$ $\langle$error-db#(;pdb$_2$)$\rangle$ pdb$_0$ = pdb$_2$
$\wedge$ $\langle$mk-db#(sent$_1$, ..., sent$_n$, rel$_1$, ..., rel$_m$;pdb$_0$)$\rangle$ pdb = pdb$_0$
$\rightarrow$ $\langle$update-E_i#(key$_i$, ent$_i$, pdb;pdb$_0$)$\rangle$ $\langle$get-E_i#(key$_i$, pdb;enti_{-0})$\rangle$
$\langle$mk-db#(sent$_1$, ..., sent$_i$ $-_{E_i}$ enti_{-0} $+_{E_i}$ ent$_i$,
..., sent$_n$,
rel$_1$, ..., rel$_m$;pdb$_2$)$\rangle$ pdb$_0$ = pdb$_2$

Proof: At first we eliminate the implication. This yields the following goal:

$\langle$rs#(pdb)$\rangle$ true,
[update-E_i#(key$_i$, ent$_i$, pdb;pdb$_0$)]pdb$_0$ = pdb$_3$,
$\langle$mk-db#(sent$_1$, ..., sent$_n$, rel$_1$, ..., rel$_m$;pdb$_0$)$\rangle$ pdb = pdb$_0$
$\vdash$
$\langle$get-E_i#(key$_i$, pdb;enti_{-0})$\rangle$
$\langle$mk-db#(sent$_1$, ..., sent$_i$ $-_{E_i}$ enti_{-0} $+_{E_i}$ ent$_i$,
..., sent$_n$,
rel$_1$, ..., rel$_m$;pdb$_2$)$\rangle$ pdb$_3$ = pdb$_2$,
$\langle$error-db#(;pdb$_2$)$\rangle$ pdb$_3$ = pdb$_2$

Now we call the procedure *get-E_i#* and then we unfold the procedure *update-E_i#*. After doing so we conclude by using the specification *coded-set* that

$$\text{sent}_i \;-_{E_i}\; \text{sel-E}_i(\text{key}_i,\text{p-ent-E}_i(\text{pdb})) \;+_{E_i}\; \text{ent}_i \neq \text{error-E}_i$$

We get the following goal:

$\langle$rs#(pdb)$\rangle$ true,
$\langle$mk-db#(sent$_1$, ..., sent$_n$, rel$_1$, ..., rel$_m$;pdb$_0$)$\rangle$ pdb = pdb$_0$
pdb $\neq$ p-error-db,

key-E_i(ent_i) = key_i, ent_i ¬ error-E_i,
$sent_i$ $-_{E_i}$ sel-E_i(key_i,p-ent-E_i(pdb)) $+_{E_i}$ ent_i ≠ error-E_i,
pdb_3 = p-mk-db(p-ent-E_1(pdb), ..., $sent_i$ $-_{E_i}$ sel-E_i(key_i,p-ent-E_i(pdb)) $+_{E_i}$ ent_i,
..., p-ent-E_n(pdb),
p-R_1(pdb), ..., p-R_m(pdb))
⊢
⟨mk-db#($sent_1$, ..., $sent_i$ $-_{E_i}$ ent$i$-$_0$ $+_{E_i}$ ent_i,
..., $sent_n$,
rel_1, ..., rel_m;pdb_2)⟩ pdb_3 = pdb_2,

We close this goal by unfolding *mk-db#* on the left side and then on the right side. In addition we use the lemmas from the proof of termination.

1.
⟨legal-R_j#(rel_j, $sent_i$, $sent_{r_{j2}}$; b)⟩ b = tt,
key-E_i(ent_i) = key-E_i(ent$i$-$_1$),
$sent_i$ $-_{E_i}$ ent$i$-$_1$ $+_{E_i}$ ent_i ≠ errorset-E_i,
⊢
⟨legal-R_j#(rel_j, $sent_i$ $-_{E_i}$ ent$i$-$_1$ $+_{E_i}$ ent_i, $sent_{r_{j2}}$;b)⟩ b = tt

2.
⟨legal-R_k#(rel_k, $sent_{r_{k1}}$, $sent_i$;b)⟩ b = tt,
key-E_i(ent_i) = key-E_i(ent$i$-$_1$),
$sent_i$ $-_{E_i}$ ent$i$-$_1$ $+_{E_i}$ ent_i ≠ errorset-E_i,
⊢
⟨legal-R_k#(rel_k, $sent_{r_{k1}}$, $sent_i$ $-_{E_i}$ ent$i$-$_1$ $+_{E_i}$ ent_i ;b)⟩ b = tt

7. The behavior of *R#*, *est-R#* and *rel-R#* on defined databases.

⟨rs#(pdb)⟩ true
⊢
¬ ⟨error-db#(;pdb_0)⟩ pdb = pdb_0
∧ ⟨mk-db#($sent_1$, ..., $sent_n$,
rel_1, ..., rel_m;pdb_0)⟩ pdb = pdb_0
∧ entr_{i1}-$_1$ ≠ error-$E_{r_{i1}}$
∧ entr_{i2}-$_2$ ≠ error-$E_{r_{i2}}$
→ (⟨R_i#(pdb, entr_{i1}-$_1$, entr_{i2}-$_2$;b)⟩ b = tt
↔ mk-R_i(key-$E_{r_{i1}}$(entr_{i1}-$_1$), key-$E_{r_{i2}}$(entr_{i2}-$_2$)) in-R_i rel_i)
∧ ⟨est-R_i#(pdb, entr_{i1}-$_1$, entr_{i2}-$_2$;pdb_0)⟩
⟨mk-db#($sent_1$, ..., $sent_n$,
rel_1, ..., rel_i $+_{R_i}$ mk-R_i(key-$E_{r_{i1}}$(entr_{i1}-$_1$), key-$E_{r_{i2}}$(entr_{i2}-$_2$)),
..., rel_m;pdb_2)⟩ pdb_0 = pdb_2
∧ ⟨rel-R_i#(pdb, entr_{i1}-$_1$, entr_{12}-$_2$;pdb_0)⟩
⟨mk-db#($sent_1$, ..., $sent_n$,
rel_1, ..., rel_i $-_{R_i}$ mk-R_i(key-$E_{r_{i1}}$(entr_{i1}-$_1$), key-$E_{r_{i2}}$(entr_{i2}-$_2$)),
..., rel_m;pdb_2)⟩ pdb_0 = pdb_2

Proof: At first we normalize this goal. This yields four sub goals. The sub goals three and four can be closed by trivial unfold steps.

1.

$$
\begin{array}{l}
\langle \mathrm{rs\#(pdb)} \rangle\, \mathrm{true}, \\
\langle \mathrm{mk\text{-}db\#(sent_1, \ldots, sent}_n, \\
\qquad\qquad \mathrm{rel_1, \ldots, rel}_m\mathrm{;pdb_0)} \rangle\, \mathrm{pdb = pdb_0}, \\
\mathrm{ent}r_{i1}\text{-}_1 \neq \mathrm{error\text{-}E}_{r_{i1}}, \\
\mathrm{ent}r_{i2}\text{-}_2 \neq \mathrm{error\text{-}E}_{r_{i2}} \\
\vdash \\
\langle \mathrm{rel\text{-}R}_i\mathrm{\#(pdb, ent}r_{i1}\text{-}_1, \mathrm{ent}r_{12}\text{-}_2\mathrm{;pdb_0)} \rangle \\
\langle \mathrm{mk\text{-}db\#(sent_1, \ldots, sent}_n, \\
\qquad\qquad \mathrm{rel_1, \ldots, rel}_i \;\text{-}_{R_i}\; \mathrm{mk\text{-}R}_i\mathrm{(key\text{-}E}_{r_{i1}}\mathrm{(ent}r_{i1}\text{-}_1), \mathrm{key\text{-}E}_{r_{i2}}\mathrm{(ent}r_{i2}\text{-}_2)), \\
\qquad\qquad \ldots, \mathrm{rel}_m\mathrm{;pdb_2)} \rangle\, \mathrm{pdb_0 = pdb_2},\ \langle \mathrm{error\text{-}db\#(;pdb_0)} \rangle\, \mathrm{pdb = pdb_0}
\end{array}
$$

2.

$$
\begin{array}{l}
\langle \mathrm{rs\#(pdb)} \rangle\, \mathrm{true}, \\
\langle \mathrm{mk\text{-}db\#(sent_1, \ldots, sent}_n, \\
\qquad\qquad \mathrm{rel_1, \ldots, rel}_m\mathrm{;pdb_0)} \rangle\, \mathrm{pdb = pdb_0}, \\
\mathrm{ent}r_{i1}\text{-}_1 \neq \mathrm{error\text{-}E}_{r_{i1}}, \\
\mathrm{ent}r_{i2}\text{-}_2 \neq \mathrm{error\text{-}E}_{r_{i2}} \\
\vdash \\
\langle \mathrm{est\text{-}R}_i\mathrm{\#(pdb, ent}r_{i1}\text{-}_1, \mathrm{ent}r_{i2}\text{-}_2\mathrm{;pdb_0)} \rangle \\
\langle \mathrm{mk\text{-}db\#(sent_1, \ldots, sent}_n, \\
\qquad\qquad \mathrm{rel_1, \ldots, rel}_i +_{R_i} \mathrm{mk\text{-}R}_i\mathrm{(key\text{-}E}_{r_{i1}}\mathrm{(ent}r_{i1}\text{-}_1), \mathrm{key\text{-}E}_{r_{i2}}\mathrm{(ent}r_{i2}\text{-}_2)), \\
\qquad\qquad \ldots, \mathrm{rel}_m\mathrm{;pdb_2)} \rangle\, \mathrm{pdb_0 = pdb_2} \\
\langle \mathrm{error\text{-}db\#(;pdb_0)} \rangle\, \mathrm{pdb = pdb_0}
\end{array}
$$

3.

$$
\begin{array}{l}
\langle \mathrm{rs\#(pdb)} \rangle\, \mathrm{true}, \\
\langle \mathrm{mk\text{-}db\#(sent_1, \ldots, sent}_n, \\
\qquad\qquad \mathrm{rel_1, \ldots, rel}_m\mathrm{;pdb_0)} \rangle\, \mathrm{pdb = pdb_0}, \\
\mathrm{ent}r_{i1}\text{-}_1 \neq \mathrm{error\text{-}E}_{r_{i1}}, \\
\mathrm{ent}r_{i2}\text{-}_2 \neq \mathrm{error\text{-}E}_{r_{i2}}, \\
\mathrm{mk\text{-}R}_i\mathrm{(key\text{-}E}_{r_{i1}}\mathrm{(ent}r_{i1}\text{-}_1), \mathrm{key\text{-}E}_{r_{i2}}\mathrm{(ent}r_{i2}\text{-}_2))\ \mathrm{in\text{-}R}_i\ \mathrm{rel}_i \\
\vdash \\
\langle \mathrm{R}_i\mathrm{\#(pdb, ent}r_{i1}\text{-}_1, \mathrm{ent}r_{i2}\text{-}_2\mathrm{;b)} \rangle\, \mathrm{b = tt}, \\
\langle \mathrm{error\text{-}db\#(;pdb_0)} \rangle\, \mathrm{pdb = pdb_0}
\end{array}
$$

4.

$$
\begin{array}{l}
\langle \mathrm{rs\#(pdb)} \rangle\, \mathrm{true}, \\
\langle \mathrm{R}_i\mathrm{\#(pdb, ent}r_{i1}\text{-}_1, \mathrm{ent}r_{i2}\text{-}_2\mathrm{;b)} \rangle\, \mathrm{b = tt}, \\
\langle \mathrm{mk\text{-}db\#(sent_1, \ldots, sent}_n, \\
\qquad\qquad \mathrm{rel_1, \ldots, rel}_m\mathrm{;pdb_0)} \rangle\, \mathrm{pdb = pdb_0}, \\
\mathrm{ent}r_{i1}\text{-}_1 \neq \mathrm{error\text{-}E}_{r_{i1}}, \\
\mathrm{ent}r_{i2}\text{-}_2 \neq \mathrm{error\text{-}E}_{r_{i2}}, \\
\neg\ \mathrm{mk\text{-}R}_i\mathrm{(key\text{-}E}_{r_{i1}}\mathrm{(ent}r_{i1}\text{-}_1), \mathrm{key\text{-}E}_{r_{i2}}\mathrm{(ent}r_{i2}\text{-}_2))\ \mathrm{in\text{-}R}_i\ \mathrm{rel}_i \\
\vdash \\
\langle \mathrm{error\text{-}db\#(;pdb_0)} \rangle\, \mathrm{pdb = pdb_0}
\end{array}
$$

Working on the first goal, at first we call the procedure *error-db#* then we unfold the first diamond on the right side. This yields the following result:

$$
\begin{array}{l}
\langle \mathrm{rs\#(pdb)} \rangle\, \mathrm{true}, \\
\langle \mathrm{mk\text{-}db\#(sent_1, \ldots, sent}_n, \\
\qquad\qquad \mathrm{rel_1, \ldots, rel}_m\mathrm{;pdb_0)} \rangle\, \mathrm{pdb = pdb_0}, \\
\mathrm{ent}r_{i1}\text{-}_1 \neq \mathrm{error\text{-}E}_{r_{i1}},
\end{array}
$$

$\text{entr}_{i2\text{-}2} \neq \text{error-E}_{r_{i2}}$,
pdb $\neq$ p-error-db,
$\text{rel}i\text{-}_0 = \text{p-R}_i(\text{pdb}) \text{-}_{R_i} \text{mk-R}_i(\text{key-E}_{r_{i1}}(\text{entr}_{i1\text{-}1}), \text{key-E}_{r_{i2}}(\text{entr}_{i2\text{-}2}))$
$\vdash$
$\langle$mk-db#($\text{sent}_1, \ldots, \text{sent}_n$,
$\text{rel}_1, \ldots, \text{rel}_i \text{-}_{R_i} \text{mk-R}_i(\text{key-E}_{r_{i1}}(\text{entr}_{i1\text{-}1}), \text{key-E}_{r_{i2}}(\text{entr}_{i2\text{-}2}))$,
$\ldots, \text{rel}_m;\text{pdb}_2)\rangle$
$\text{p-mk-db}(\text{p-ent-E}_1(\text{pdb}), \ldots, \text{p-ent-E}_n(\text{pdb}),$
$\text{p-R}_1(\text{pdb}), \ldots, \text{rel}i\text{-}_0, \ldots, \text{p-R}_m(\text{pdb})) = \text{pdb}_2$

We close this goal by unfolding *mk-db#* on the left side and then on the right side. In addition we use the lemma from the termination proof of *rel-*R_i*#*.

$\langle \text{legal-R}_i\#(\text{rel}_i, \text{sent}_{r_{i1}}, \text{sent}_{r_{i2}}; \text{b})\rangle\ \text{b} = \text{tt}$,
$\vdash$
$\langle \text{legal-R}_i\#(\text{rel}_i \text{-}_{R_i} \text{k}_i\ ,_{r_{i1}}, \text{sent}_{r_{i2}};\text{b})\rangle\ \text{b} = \text{tt}$

Working on the second goal, like before we unfold the procedures *error-db#* and *est-*R_i*#*. For closing the goal we also need the lemma from the termination proof of *est-*R_i*#*. But for the error case we need one lemma more:

$\text{sent}_{r_{i1}} \neq \text{errorset-E}_{r_{i1}}, \text{sent}_{r_{i2}} \neq \text{errorset-E}_{r_{i2}}$,
$\text{sel-ent-E}_{r_{j1}}(\text{fst-R}_i(\text{k}_i), \text{sent}_{r_{i1}}) = \text{error-E}_{r_{i1}}$
$\vee\ \text{sel-ent-E}_{r_{j2}}(\text{snd-R}_i(\text{k}_i), \text{sent}_{r_{i2}}) = \text{error-E}_{r_{i2}}$
$\vdash$
$\langle \text{legal-R}_i\#(\text{rel}_i +_{R_i} \text{k}_i\ ,_{r_{i1}}, \text{sent}_{r_{i2}};\text{b})\rangle\ \text{b} = \text{ff}$

This lemma can be proved by induction like the lemmas before.

8. The following goals deals with error propagation. Therefore the proof is trivial. Only the last case looks more complex but after normalization it is easy too.

1.

$\vdash \langle \text{error-db}\#(;\text{pdb}_0)\rangle\ \langle \text{ent-E}_i\#(\text{pdb}_0;\text{sent}i\text{-}_0)\rangle\ \text{sent}i\text{-}_0 = \text{errorset-E}_i$

2.

$\vdash \langle \text{error-db}\#(;\text{pdb}_0)\rangle\ \langle \text{put-E}_i\#(\text{ent}_i, \text{pdb}_0;\text{pdb}_2)\rangle\ \langle \text{error-db}\#(;\text{pdb}_3)\rangle\ \text{pdb}_2 = \text{pdb}_3$

3.

$\vdash \langle \text{error-db}\#(;\text{pdb}_0)\rangle\ \langle \text{del-E}_i\#(\text{ent}_i, \text{pdb}_0;\text{pdb}_2)\rangle\ \langle \text{error-db}\#(;\text{pdb}_3)\rangle\ \text{pdb}_2 = \text{pdb}_3$

4.

$\langle \text{rs}\#(\text{pdb})\rangle$ true
$\vdash$
$\langle \text{error-db}\#(;\text{pdb}_0)\rangle \text{pdb} = \text{pdb}_0$
$\vee\ \text{entr}_{11\text{-}1} = \text{error-E}_{r_{11}}$
$\vee\ \text{entr}_{12\text{-}2} = \text{error-E}_{r_{12}}$
$\rightarrow \neg\ \langle \text{R}_1\#(\text{pdb}, \text{entr}_{11\text{-}1}, \text{entr}_{12\text{-}2};\text{b})\rangle \text{b} = \text{tt}$
$\wedge\ \langle \text{est-R}_1\#(\text{pdb}, \text{entr}_{11\text{-}1}, \text{entr}_{12\text{-}2};\text{pdb}_0)\rangle\ \langle \text{error-db}\#(;\text{pdb}_2)\rangle \text{pdb}_0 = \text{pdb}_2$
$\wedge\ \langle \text{rel-R}_1\#(\text{pdb}, \text{entr}_{11\text{-}1}, \text{entr}_{12\text{-}2};\text{pdb}_0)\rangle\ \langle \text{error-db}\#(;\text{pdb}_2)\rangle \text{pdb}_0 = \text{pdb}_2$

The Condition for the Restriction (iiii)

Before we are going to prove the condition. We present the restriction which is generated in an uniform way by translating the generating axioms of the export specification.

```
uniform_rs#(pdb1)
begin
   var sent1- = ?, ..., sentn-1 = ?, rel1- = ?, ..., relm-1 = ?, pdb0 = pdb1 in
   begin
      mk-db#(sent1-1, ..., sentn-1, rel1-1, ..., relm-1; pdb0);
      if pdb0 = pdb1 then skip else
        var pdb = pdb1 in
        begin
           error-db#(; pdb);
           if pdb = pdb1 then skip else abort
        end
   end
end
```

$\langle$rs#(pdb_1)$\rangle$ true $\vdash$ $\langle$uniform_rs#(pdb_1)$\rangle$ true

Proof: Like in the case of the entity implementation we unfold the right side and then the left side until we get the following two sub goals:

1.

$\vdash$

$\exists$ sent1-$_1$, ..., sent$n$-$_1$, rel1-$_1$, ..., rel$m$-$_1$.

```
⟨begin
    mk-db#(sent1-1, ..., sentn-1, rel1-1, ..., relm-1; pdb2);
    if pdb2 = p-error-db then skip else
      var pdb = p-error-db in
      begin
         error-db#(; pdb);
         if pdb = p-error-db then skip else abort
      end
end ⟩ true
```

2.

$\langle$legal-R_1#(p-R_1(pdb_1), p-ent-$E_{r_{11}}$(pdb_1), p-ent-$E_{r_{12}}$(pdb_1); b_0)$\rangle$ b_0 = tt,

$\vdots$

$\langle$legal-R_m#(p-R_m(pdb_1), p-ent-$E_{r_{m1}}$(pdb_1), p-ent-$E_{r_{m2}}$(pdb_1); b_0)$\rangle$ b_0 = tt,

$pdb_1 \neq$ p-error-db, p-ent-E_1(pdb_1) $\neq$ errorset-E_1, ..., p-ent-E_n(pdb_1) $\neq$ errorset-E_n

$\vdash$

$\exists$ sent1-$_1$, ..., sent$n$-$_1$, rel1-$_1$, ..., rel$m$-$_1$.

```
⟨begin
    mk-db#(sent1-1, ..., sentn-1, rel1-1, ..., relm-1; pdb2);
    if pdb2 = pdb1 then skip else
      var pdb = pdb1 in
      begin
         error-db#(; pdb);
         if pdb = pdb1 then skip else abort
      end
end ⟩ true
```

In the first case we instantiate *sent*1-$_1$, ..., *sentn*-$_1$, *rel1*-$_1$, ..., *relm*-$_1$ with *errorset*-E_1, ..., *errorset*-E_n, rel_1, ..., rel_m. In the second case we instantiate *sent*1-$_1$, ..., *sentn*-$_1$, *rel1*-$_1$, ..., *relm*-$_1$ with *p-ent*-$E_1(pdb_1)$, ..., *p-ent*-$E_n(pdb_1)$, *p*-$R_1(pdb_1)$, ..., (*p*-$R_m(pdb_1)$. Both goals can now be closed by unfolding the right side.

This completes the consistency proof of the database specification.

Chapter 5

Conclusion

In the previous chapters we have presented a method for proving consistency of specifications. The idea of a prototyped implementation and a correctness proof seems to be an adequate method for doing this job. But there exists at least one problem:

> We need an explicit formal proof for every instantiation, if we will guarantee consistency. Because the transformation is uniform for each E/R-diagram this can be reduced to a generalized proof using the templates. We have such a sketched proof in the previous chapter. In principle this yields another problem. We have to guarantee that the used translation program generates for arbitrary E/R-diagrams a right instantiation of the schemes. A more complex problem, which requires a formal verification of the transformation program.

To overcome the last problem we have two possibilities. First we regard the correctness of the translation program as a parameter of the consistency proof. This means the specification is consistent modulo the correctness of the translation. Second, the translation program produces not only the specifications and implementations but also the proofs or proof plans. This method has the advantage of independence between correctness of the transformation program and the generated modular system in addition. Because each concrete modular system could be completely proved with minimal expense if a proof plan exists or we only need a proof checker if a complete proof is produced. For the moment we estimate the expense of developing a proof plan.

In the appendix we present the instantiation of the schemes for the example Cardiac-Catheterisation. If we compare the size of the semi formal description (a half page) and the size of the translation (approximately 80 pages) we must conclude that semi formal methods cannot be replaced by formal methods in human communication processes without losing clarity. But altogether formal descriptions are necessary for formal program development, and combined with a prototyped implementation they are an useful method. And it would be more practicable if we can use partial functions and abbreviations like strictness clauses, because almost half of the axioms deal with error propagation. Such abbreviations don't reduce the number of proof obligations but the number of written axioms. Thereby we can reduce the size of the specification to a more readable size, and focus the view on the central parts of the specification.

Appendix A

The Example Cardiac-Catheterisation

As an appendix we present the instantiation of the the described schemes for the example Cardiac-Catheterisation (see chapter 1).

A.1 The Attributes

A.1.1 The Specifications

AttributesParam

An instantiation of the same name scheme.

AttributesParam =
specification
sorts
normal-patids, normal-names, normal-tsex, normal-dates, normal-place,
normal-bearers, normal-adr, normal-physician, normal-physical-data,
normal-wards, normal-rooms, normal-doctorids, normal-trank, normal-time,
normal-roles, normal-ccid, normal-text, normal-curves, normal-film,
normal-findingsids, normal-tfindings, normal-letter;
predicates
$\cdot \ll_{n-findingsids} \cdot$: normal-findingsids $\times$ normal-findingsids;
$\cdot \ll_{n-ccid} \cdot$: normal-ccid $\times$ normal-ccid;
$\cdot \ll_{n-doctorids} \cdot$: normal-doctorids $\times$ normal-doctorids;
$\cdot \ll_{n-patids} \cdot$: normal-patids $\times$ normal-patids;
variables
v-n-patids-$_2$, v-n-patids-$_1$, v-n-patids: normal-patids;
v-n-names: normal-names;
v-n-tsex: normal-tsex;
v-n-dates: normal-dates;
v-n-place: normal-place;
v-n-bearers: normal-bearers;
v-n-adr: normal-adr;
v-n-physician: normal-physician;
v-n-physical-data: normal-physical-data;
v-n-wards: normal-wards;
v-n-rooms: normal-rooms;
v-n-doctorids-$_2$, v-n-doctorids-$_1$, v-n-doctorids: normal-doctorids;
v-n-trank: normal-trank;
v-n-time: normal-time;

v-n-roles: normal-roles;
v-n-ccid-$_2$, v-n-ccid-$_1$, v-n-ccid: normal-ccid;
v-n-text: normal-text;
v-n-curves: normal-curves;
v-n-film: normal-film;
v-n-findingsids-$_2$, v-n-findingsids-$_1$, v-n-findingsids: normal-findingsids;
v-n-tfindings: normal-tfindings;
v-n-letter: normal-letter;

axioms

$\neg$ v-n-findingsids $\ll_{n-findingsids}$ v-n-findingsids,
v-n-findingsids $\ll_{n-findingsids}$ v-n-findingsids-$_1$
$\wedge$ v-n-findingsids-$_1$ $\ll_{n-findingsids}$ v-n-findingsids-$_2$
$\rightarrow$ v-n-findingsids $\ll_{n-findingsids}$ v-n-findingsids-$_2$,
v-n-findingsids $\ll_{n-findingsids}$ v-n-findingsids-$_1$
$\vee$ v-n-findingsids = v-n-findingsids-$_1$
$\vee$ v-n-findingsids-$_1$ $\ll_{n-findingsids}$ v-n-findingsids,
$\neg$ v-n-ccid $\ll_{n-ccid}$ v-n-ccid,
v-n-ccid $\ll_{n-ccid}$ v-n-ccid-$_1$
$\wedge$ v-n-ccid-$_1$ $\ll_{n-ccid}$ v-n-ccid-$_2$
$\rightarrow$ v-n-ccid $\ll_{n-ccid}$ v-n-ccid-$_2$,
v-n-ccid $\ll_{n-ccid}$ v-n-ccid-$_1$
$\vee$ v-n-ccid = v-n-ccid-$_1$
$\vee$ v-n-ccid-$_1$ $\ll_{n-ccid}$ v-n-ccid,
$\neg$ v-n-doctorids $\ll_{n-doctorids}$ v-n-doctorids,
v-n-doctorids $\ll_{n-doctorids}$ v-n-doctorids-$_1$
$\wedge$ v-n-doctorids-$_1$ $\ll_{n-doctorids}$ v-n-doctorids-$_2$
$\rightarrow$ v-n-doctorids $\ll_{n-doctorids}$ v-n-doctorids-$_2$,
v-n-doctorids $\ll_{n-doctorids}$ v-n-doctorids-$_1$
$\vee$ v-n-doctorids = v-n-doctorids-$_1$
$\vee$ v-n-doctorids-$_1$ $\ll_{n-doctorids}$ v-n-doctorids,
$\neg$ v-n-patids $\ll_{n-patids}$ v-n-patids,
v-n-patids $\ll_{n-patids}$ v-n-patids-$_1$
$\wedge$ v-n-patids-$_1$ $\ll_{n-patids}$ v-n-patids-$_2$
$\rightarrow$ v-n-patids $\ll_{n-patids}$ v-n-patids-$_2$,
v-n-patids $\ll_{n-patids}$ v-n-patids-$_1$
$\vee$ v-n-patids = v-n-patids-$_1$
$\vee$ v-n-patids-$_1$ $\ll_{n-patids}$ v-n-patids

end specification

Attributes

An instantiation of the same name scheme.

Attributes =
generic data specification
parameter AttributesParam
patids = error-patids
| undef-patids
| copy-patids (get-patids : normal-patids)
with copy-patids-prd ;
names = error-names
| undef-names
| copy-names (get-names : normal-names)
with copy-names-prd ;
tsex = error-tsex
| undef-tsex
| copy-tsex (get-tsex : normal-tsex)

```
        with copy-tsex-prd ;
dates = error-dates
        | undef-dates
        | copy-dates (get-dates : normal-dates)
        with copy-dates-prd ;
place = error-place
        | undef-place
        | copy-place (get-place : normal-place)
        with copy-place-prd ;
bearers = error-bearers
        | undef-bearers
        | copy-bearers (get-bearers : normal-bearers)
        with copy-bearers-prd ;
adr = error-adr
        | undef-adr
        | copy-adr (get-adr : normal-adr)
        with copy-adr-prd ;
physician = error-physician
        | undef-physician
        | copy-physician (get-physician : normal-physician)
        with copy-physician-prd ;
physical-data = error-physical-data
        | undef-physical-data
        | copy-physical-data (get-physical-data : normal-physical-data)
        with copy-physical-data-prd ;
wards = error-wards
        | undef-wards
        | copy-wards (get-wards : normal-wards)
        with copy-wards-prd ;
rooms = error-rooms
        | undef-rooms
        | copy-rooms (get-rooms : normal-rooms)
        with copy-rooms-prd ;
doctorids = error-doctorids
        | undef-doctorids
        | copy-doctorids (get-doctorids : normal-doctorids)
        with copy-doctorids-prd ;
trank = error-trank
        | undef-trank
        | copy-trank (get-trank : normal-trank)
        with copy-trank-prd ;
time = error-time
        | undef-time
        | copy-time (get-time : normal-time)
        with copy-time-prd ;
roles = error-roles
        | undef-roles
        | copy-roles (get-roles : normal-roles)
        with copy-roles-prd ;
ccid = error-ccid
        | undef-ccid
        | copy-ccid (get-ccid : normal-ccid)
        with copy-ccid-prd ;
text = error-text
        | undef-text
        | copy-text (get-text : normal-text)
```

```
            with copy-text-prd ;
    curves =  error-curves
              | undef-curves
              | copy-curves (get-curves : normal-curves)
              with copy-curves-prd ;
    film =  error-film
            | undef-film
            | copy-film (get-film : normal-film)
            with copy-film-prd ;
    findingsids =  error-findingsids
                   | undef-findingsids
                   | copy-findingsids (get-findingsids : normal-findingsids)
                   with copy-findingsids-prd ;
    tfindings =  error-tfindings
                 | undef-tfindings
                 | copy-tfindings (get-tfindings : normal-tfindings)
                 with copy-tfindings-prd ;
    letter =  error-letter
              | undef-letter
              | copy-letter (get-letter : normal-letter)
              with copy-letter-prd ;
    variables
        v-patids-2, v-patids-1, v-patids: patids;
        v-names: names;
        v-tsex: tsex;
        v-dates: dates;
        v-place: place;
        v-bearers: bearers;
        v-adr: adr;
        v-physician: physician;
        v-physical-data: physical-data;
        v-wards: wards;
        v-rooms: rooms;
        v-doctorids-2, v-doctorids-1, v-doctorids: doctorids;
        v-trank: trank;
        v-time: time;
        v-roles: roles;
        v-ccid-2, v-ccid-1, v-ccid: ccid;
        v-text: text;
        v-curves: curves;
        v-film: film;
        v-findingsids-2, v-findingsids-1, v-findingsids: findingsids;
        v-tfindings: tfindings;
        v-letter: letter;
end generic data specification
```

OrderedAttributes

An instantiation of the same name scheme.

OrderedAttributes =
enrich Attributes **with**
  predicates

$\cdot \ll_{findingsids} \cdot$	:	findingsids × findingsids;
$\cdot \ll_{ccid} \cdot$	:	ccid × ccid;
$\cdot \ll_{doctorids} \cdot$	:	doctorids × doctorids;
$\cdot \ll_{patids} \cdot$	:	patids × patids;

axioms

error-findingsids $\ll_{findingsids}$ undef-findingsids,
copy-findingsids-prd(v-findingsids) $\rightarrow$ undef-findingsids $\ll_{findingsids}$ v-findingsids,
copy-findingsids(v-n-findingsids) $\ll_{findingsids}$ copy-findingsids(v-n-findingsids-$_1$)
$\leftrightarrow$ v-n-findingsids $\ll_{n-findingsids}$ v-n-findingsids-$_1$,
error-ccid $\ll_{ccid}$ undef-ccid,
copy-ccid-prd(v-ccid) $\rightarrow$ undef-ccid $\ll_{ccid}$ v-ccid,
copy-ccid(v-n-ccid) $\ll_{ccid}$ copy-ccid(v-n-ccid-$_1$) $\leftrightarrow$ v-n-ccid $\ll_{n-ccid}$ v-n-ccid-$_1$,
error-doctorids $\ll_{doctorids}$ undef-doctorids,
copy-doctorids-prd(v-doctorids) $\rightarrow$ undef-doctorids $\ll_{doctorids}$ v-doctorids,
copy-doctorids(v-n-doctorids) $\ll_{doctorids}$ copy-doctorids(v-n-doctorids-$_1$)
$\leftrightarrow$ v-n-doctorids $\ll_{n-doctorids}$ v-n-doctorids-$_1$,
error-patids $\ll_{patids}$ undef-patids,
copy-patids-prd(v-patids) $\rightarrow$ undef-patids $\ll_{patids}$ v-patids,
copy-patids(v-n-patids) $\ll_{patids}$ copy-patids(v-n-patids-$_1$)
$\leftrightarrow$ v-n-patids $\ll_{n-patids}$ v-n-patids-$_1$,
$\neg$ v-findingsids $\ll_{findingsids}$ v-findingsids,
v-findingsids $\ll_{findingsids}$ v-findingsids-$_1$ $\wedge$ v-findingsids-$_1$ $\ll_{findingsids}$ v-findingsids-$_2$
$\rightarrow$ v-findingsids $\ll_{findingsids}$ v-findingsids-$_2$,
v-findingsids $\ll_{findingsids}$ v-findingsids-$_1$
$\vee$ v-findingsids = v-findingsids-$_1$
$\vee$ v-findingsids-$_1$ $\ll_{findingsids}$ v-findingsids,
$\neg$ v-ccid $\ll_{ccid}$ v-ccid,
v-ccid $\ll_{ccid}$ v-ccid-$_1$ $\wedge$ v-ccid-$_1$ $\ll_{ccid}$ v-ccid-$_2$ $\rightarrow$ v-ccid $\ll_{ccid}$ v-ccid-$_2$,
v-ccid $\ll_{ccid}$ v-ccid-$_1$ $\vee$ v-ccid = v-ccid-$_1$ $\vee$ v-ccid-$_1$ $\ll_{ccid}$ v-ccid,
$\neg$ v-doctorids $\ll_{doctorids}$ v-doctorids,
v-doctorids $\ll_{doctorids}$ v-doctorids-$_1$ $\wedge$ v-doctorids-$_1$ $\ll_{doctorids}$ v-doctorids-$_2$
$\rightarrow$ v-doctorids $\ll_{doctorids}$ v-doctorids-$_2$,
v-doctorids $\ll_{doctorids}$ v-doctorids-$_1$
$\vee$ v-doctorids = v-doctorids-$_1$
$\vee$ v-doctorids-$_1$ $\ll_{doctorids}$ v-doctorids,
$\neg$ v-patids $\ll_{patids}$ v-patids,
v-patids $\ll_{patids}$ v-patids-$_1$ $\wedge$ v-patids-$_1$ $\ll_{patids}$ v-patids-$_2$
$\rightarrow$ v-patids $\ll_{patids}$ v-patids-$_2$,
v-patids $\ll_{patids}$ v-patids-$_1$ $\vee$ v-patids = v-patids-$_1$ $\vee$ v-patids-$_1$ $\ll_{patids}$ v-patids

end enrich

A.1.2 The Implementation

OrderedAttribs-Attribs =
module
export OrderedAttributes
refinement
representation of operations

patids≪#	implements	$\ll_{patids}$;
doctorids≪#	implements	$\ll_{doctorids}$;
ccid≪#	implements	$\ll_{ccid}$;
findingsids≪#	implements	$\ll_{findingsids}$;

import Attributes

procedures

patids≪#	(patids, patids)	: bool;
doctorids≪#	(doctorids, doctorids)	: bool;
ccid≪#	(ccid, ccid)	: bool;
findingsids≪#	(findingsids, findingsids)	: bool;

```
    variables b: bool;

    implementation


patids≪#(v-patids, v-patids-1; var b)
begin
  if copy-patids-prd(v-patids) ∧ copy-patids-prd(v-patids-1) then
    if get-patids(v-patids) ≪n-patids get-patids(v-patids-1) then b := tt else b := ff
  else
    if (v-patids = error-patids ∧ v-patids-1 ≠ error-patids)
      ∨ (v-patids = undef-patids ∧ copy-patids-prd(v-patids-1)) then
      b := tt
    else
      b := ff
end


doctorids≪#(v-doctorids, v-doctorids-1; var b)
begin
  if copy-doctorids-prd(v-doctorids) ∧ copy-doctorids-prd(v-doctorids-1) then
    if get-doctorids(v-doctorids) ≪n-doctorids get-doctorids(v-doctorids-1) then
      b := tt else b := ff
  else
    if (v-doctorids = error-doctorids ∧ v-doctorids-1 ≠ error-doctorids)
      ∨ (v-doctorids = undef-doctorids ∧ copy-doctorids-prd(v-doctorids-1)) then
      b := tt
    else
      b := ff
end


ccid≪#(v-ccid, v-ccid-1; var b)
begin
  if copy-ccid-prd(v-ccid) ∧ copy-ccid-prd(v-ccid-1) then
    if get-ccid(v-ccid) ≪n-ccid get-ccid(v-ccid-1) then b := tt else b := ff
  else
    if (v-ccid = error-ccid ∧ v-ccid-1 ≠ error-ccid)
      ∨ (v-ccid = undef-ccid ∧ copy-ccid-prd(v-ccid-1)) then
      b := tt
    else
      b := ff
end


findingsids≪#(v-findingsids, v-findingsids-1; var b)
begin
  if copy-findingsids-prd(v-findingsids) ∧ copy-findingsids-prd(v-findingsids-1) then
    if get-findingsids(v-findingsids) ≪n-findingsids get-findingsids(v-findingsids-1) then
      b := tt
    else
      b := ff
  else
```

```
    if (v-findingsids = error-findingsids ∧ v-findingsids-_1 ≠ error-findingsids)
      ∨ (v-findingsids = undef-findingsids ∧ copy-findingsids-prd(v-findingsids-_1)) then
      b := tt
    else
      b := ff
end
```

A.2 The Entities

A.2.1 Patient

The Specifications

The Specification prePatient An instantiation of the scheme preEntity_i.

```
 prePatient =
data specification
   using OrderedAttributes
   pre-patient =  p-mk-patient (p-patient_id : patids,
                               p-name : names,
                               p-sex : tsex,
                               p-birthdate : dates,
                               p-birthplace : place,
                               p-costebearer : bearers,
                               p-address : adr,
                               p-famdoctor : physician,
                               p-physicaldata : physical-data,
                               p-ward : wards,
                               p-room : rooms)

                 | p-error-patient
                 ;
   p-keysort-patient = p-mkkey-patient (k-patient_id : patids);
   variables
        pent1-_1, pent_1: pre-patient;
        pkey1-_1, pkey_1: p-keysort-patient;
end data specification
```

The Specification Patient An instantiation of the scheme Entity_i.

```
 Patient =
enrich OrderedAttributes with
   sorts patient, keysort-patient;
   constants error-patient : patient;
   functions
       create-patient  :  patids × names × tsex × dates
                          × place × bearers × adr × physician
                          × physical-data × wards × rooms      →  patient  ;
       patient_id      :  patient                              →  patids   ;
       name            :  patient                              →  names    ;
       sex             :  patient                              →  tsex     ;
       birthdate       :  patient                              →  dates    ;
       birthplace      :  patient                              →  place    ;
```

costebearer	:	patient	$\rightarrow$	bearers	;
address	:	patient	$\rightarrow$	adr	;
famdoctor	:	patient	$\rightarrow$	physician	;
physicaldata	:	patient	$\rightarrow$	physical-data	;
ward	:	patient	$\rightarrow$	wards	;
room	:	patient	$\rightarrow$	rooms	;
set-patient_id	:	patient $\times$ patids	$\rightarrow$	patient	;
set-name	:	patient $\times$ names	$\rightarrow$	patient	;
set-sex	:	patient $\times$ tsex	$\rightarrow$	patient	;
set-birthdate	:	patient $\times$ dates	$\rightarrow$	patient	;
set-birthplace	:	patient $\times$ place	$\rightarrow$	patient	;
set-costebearer	:	patient $\times$ bearers	$\rightarrow$	patient	;
set-address	:	patient $\times$ adr	$\rightarrow$	patient	;
set-famdoctor	:	patient $\times$ physician	$\rightarrow$	patient	;
set-physicaldata	:	patient $\times$ physical-data	$\rightarrow$	patient	;
set-ward	:	patient $\times$ wards	$\rightarrow$	patient	;
set-room	:	patient $\times$ rooms	$\rightarrow$	patient	;
key-patient	:	patient	$\rightarrow$	keysort-patient	;
mkkey-patient	:	patids	$\rightarrow$	keysort-patient	;

predicates . $\ll_{key-patient}$. : keysort-patient $\times$ keysort-patient;

variables

ent_1: patient;
$key1\text{-}_2$, $key1\text{-}_1$, key_1: keysort-patient;
a_{10}, a: patids;
a_{11}, a_0: names;
a_{12}, a_1: tsex;
a_{13}, a_2: dates;
a_{14}, a_3: place;
a_{15}, a_4: bearers;
a_{16}, a_5: adr;
a_{17}, a_6: physician;
a_{18}, a_7: physical-data;
a_{19}, a_8: wards;
a_{20}, a_9: rooms;

axioms

patient **generated by** create-patient, error-patient;
keysort-patient **freely generated by** mkkey-patient;
a = undef-patids
$\vee$ a_0 = undef-names
$\vee$ a_1 = undef-tsex
$\vee$ a_2 = undef-dates
$\vee$ a_3 = undef-place
$\vee$ a_4 = undef-bearers
$\vee$ a = error-patids
$\vee$ a_0 = error-names
$\vee$ a_1 = error-tsex
$\vee$ a_2 = error-dates
$\vee$ a_3 = error-place
$\vee$ a_4 = error-bearers
$\vee$ a_5 = error-adr
$\vee$ a_6 = error-physician
$\vee$ a_7 = error-physical-data
$\vee$ a_8 = error-wards
$\vee$ a_9 = error-rooms
$\leftrightarrow$
create-patient(a, a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9) = error-patient,

create-patient(a, a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9) $\neq$ error-patient
$\rightarrow$ (create-patient(a, a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9)
= create-patient(a_{10}, a_{11}, a_{12}, a_{13}, a_{14}, a_{15}, a_{16}, a_{17}, a_{18}, a_{19}, a_{20})
$\rightarrow$ a = a_{10}
$\wedge$ $a_0 = a_{11}$
$\wedge$ $a_1 = a_{12}$
$\wedge$ $a_2 = a_{13}$
$\wedge$ $a_3 = a_{14}$
$\wedge$ $a_4 = a_{15}$
$\wedge$ $a_5 = a_{16}$
$\wedge$ $a_6 = a_{17}$
$\wedge$ $a_7 = a_{18}$
$\wedge$ $a_8 = a_{19}$
$\wedge$ $a_9 = a_{20}$)
$\wedge$ patient_id(create-patient(a, a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9)) = a
$\wedge$ name(create-patient(a, a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9)) = a_0
$\wedge$ sex(create-patient(a, a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9)) = a_1
$\wedge$ birthdate(create-patient(a, a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9)) = a_2
$\wedge$ birthplace(create-patient(a, a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9)) = a_3
$\wedge$ costebearer(create-patient(a, a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9)) = a_4
$\wedge$ address(create-patient(a, a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9)) = a_5
$\wedge$ famdoctor(create-patient(a, a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9)) = a_6
$\wedge$ physicaldata(create-patient(a, a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9)) = a_7
$\wedge$ ward(create-patient(a, a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9)) = a_8
$\wedge$ room(create-patient(a, a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9)) = a_9,
ent_1 $\neq$ error-patient
$\rightarrow$ set-patient_id(ent_1, a)
= create-patient(a,
name(ent_1),
sex(ent_1),
birthdate(ent_1),
birthplace(ent_1),
costebearer(ent_1),
address(ent_1),
famdoctor(ent_1),
physicaldata(ent_1),
ward(ent_1),
room(ent_1))
$\wedge$ set-name(ent_1, a_0)
= create-patient(patient_id(ent_1),
a_0,
sex(ent_1),
birthdate(ent_1),
birthplace(ent_1),
costebearer(ent_1),
address(ent_1),
famdoctor(ent_1),
physicaldata(ent_1),
ward(ent_1),
room(ent_1))
$\wedge$ set-sex(ent_1, a_1)
= create-patient(patient_id(ent_1),
name(ent_1),
a_1,
birthdate(ent_1),
birthplace(ent_1),

$$
\begin{aligned}
& \qquad\qquad \text{costebearer}(\text{ent}_1), \\
& \qquad\qquad \text{address}(\text{ent}_1), \\
& \qquad\qquad \text{famdoctor}(\text{ent}_1), \\
& \qquad\qquad \text{physicaldata}(\text{ent}_1), \\
& \qquad\qquad \text{ward}(\text{ent}_1), \\
& \qquad\qquad \text{room}(\text{ent}_1)) \\
\wedge\ & \text{set-birthdate}(\text{ent}_1, a_2) \\
& = \text{create-patient}(\text{patient\_id}(\text{ent}_1), \\
& \qquad\qquad \text{name}(\text{ent}_1), \\
& \qquad\qquad \text{sex}(\text{ent}_1), \\
& \qquad\qquad a_2, \\
& \qquad\qquad \text{birthplace}(\text{ent}_1), \\
& \qquad\qquad \text{costebearer}(\text{ent}_1), \\
& \qquad\qquad \text{address}(\text{ent}_1), \\
& \qquad\qquad \text{famdoctor}(\text{ent}_1), \\
& \qquad\qquad \text{physicaldata}(\text{ent}_1), \\
& \qquad\qquad \text{ward}(\text{ent}_1), \\
& \qquad\qquad \text{room}(\text{ent}_1)) \\
\wedge\ & \text{set-birthplace}(\text{ent}_1, a_3) \\
& = \text{create-patient}(\text{patient\_id}(\text{ent}_1), \\
& \qquad\qquad \text{name}(\text{ent}_1), \\
& \qquad\qquad \text{sex}(\text{ent}_1), \\
& \qquad\qquad \text{birthdate}(\text{ent}_1), \\
& \qquad\qquad a_3, \\
& \qquad\qquad \text{costebearer}(\text{ent}_1), \\
& \qquad\qquad \text{address}(\text{ent}_1), \\
& \qquad\qquad \text{famdoctor}(\text{ent}_1), \\
& \qquad\qquad \text{physicaldata}(\text{ent}_1), \\
& \qquad\qquad \text{ward}(\text{ent}_1), \\
& \qquad\qquad \text{room}(\text{ent}_1)) \\
\wedge\ & \text{set-costebearer}(\text{ent}_1, a_4) \\
& = \text{create-patient}(\text{patient\_id}(\text{ent}_1), \\
& \qquad\qquad \text{name}(\text{ent}_1), \\
& \qquad\qquad \text{sex}(\text{ent}_1), \\
& \qquad\qquad \text{birthdate}(\text{ent}_1), \\
& \qquad\qquad \text{birthplace}(\text{ent}_1), \\
& \qquad\qquad a_4, \\
& \qquad\qquad \text{address}(\text{ent}_1), \\
& \qquad\qquad \text{famdoctor}(\text{ent}_1), \\
& \qquad\qquad \text{physicaldata}(\text{ent}_1), \\
& \qquad\qquad \text{ward}(\text{ent}_1), \\
& \qquad\qquad \text{room}(\text{ent}_1)) \\
\wedge\ & \text{set-address}(\text{ent}_1, a_5) \\
& = \text{create-patient}(\text{patient\_id}(\text{ent}_1), \\
& \qquad\qquad \text{name}(\text{ent}_1), \\
& \qquad\qquad \text{sex}(\text{ent}_1), \\
& \qquad\qquad \text{birthdate}(\text{ent}_1), \\
& \qquad\qquad \text{birthplace}(\text{ent}_1), \\
& \qquad\qquad \text{costebearer}(\text{ent}_1), \\
& \qquad\qquad a_5, \\
& \qquad\qquad \text{famdoctor}(\text{ent}_1), \\
& \qquad\qquad \text{physicaldata}(\text{ent}_1), \\
& \qquad\qquad \text{ward}(\text{ent}_1), \\
& \qquad\qquad \text{room}(\text{ent}_1)) \\
\wedge\ & \text{set-famdoctor}(\text{ent}_1, a_6) \\
& = \text{create-patient}(\text{patient\_id}(\text{ent}_1),
\end{aligned}
$$

name(ent_1),
sex(ent_1),
birthdate(ent_1),
birthplace(ent_1),
costebearer(ent_1),
address(ent_1),
a_6,
physicaldata(ent_1),
ward(ent_1),
room(ent_1))
$\wedge$ set-physicaldata(ent_1, a_7)
= create-patient(patient_id(ent_1),
name(ent_1),
sex(ent_1),
birthdate(ent_1),
birthplace(ent_1),
costebearer(ent_1),
address(ent_1),
famdoctor(ent_1),
a_7,
ward(ent_1),
room(ent_1))
$\wedge$ set-ward(ent_1, a_8)
= create-patient(patient_id(ent_1),
name(ent_1),
sex(ent_1),
birthdate(ent_1),
birthplace(ent_1),
costebearer(ent_1),
address(ent_1),
famdoctor(ent_1),
physicaldata(ent_1),
a_8,
room(ent_1))
$\wedge$ set-room(ent_1, a_9)
= create-patient(patient_id(ent_1),
name(ent_1),
sex(ent_1),
birthdate(ent_1),
birthplace(ent_1),
costebearer(ent_1),
address(ent_1),
famdoctor(ent_1),
physicaldata(ent_1),
ward(ent_1),
a_9),
patient_id(error-patient) = error-patids,
name(error-patient) = error-names,
sex(error-patient) = error-tsex,
birthdate(error-patient) = error-dates,
birthplace(error-patient) = error-place,
costebearer(error-patient) = error-bearers,
address(error-patient) = error-adr,
famdoctor(error-patient) = error-physician,
physicaldata(error-patient) = error-physical-data,
ward(error-patient) = error-wards,

room(error-patient) = error-rooms,
set-patient_id(error-patient, a) = error-patient,
set-name(error-patient, a_0) = error-patient,
set-sex(error-patient, a_1) = error-patient,
set-birthdate(error-patient, a_2) = error-patient,
set-birthplace(error-patient, a_3) = error-patient,
set-costebearer(error-patient, a_4) = error-patient,
set-address(error-patient, a_5) = error-patient,
set-famdoctor(error-patient, a_6) = error-patient,
set-physicaldata(error-patient, a_7) = error-patient,
set-ward(error-patient, a_8) = error-patient,
set-room(error-patient, a_9) = error-patient,
key-patient(ent_1) = mkkey-patient(patient_id(ent_1)),
mkkey-patient(a) $\ll_{key-patient}$ mkkey-patient(a_{10}) $\leftrightarrow$ a $\ll_{patids}$ a_{10},
$\neg$ key_1 $\ll_{key-patient}$ key_1,
key_1 $\ll_{key-patient}$ key1-$_1$ $\wedge$ key1-$_1$ $\ll_{key-patient}$ key1-$_2$
$\rightarrow$ key_1 $\ll_{key-patient}$ key1-$_2$,
key_1 $\ll_{key-patient}$ key1-$_1$ $\vee$ key_1 = key1-$_1$ $\vee$ key1-$_1$ $\ll_{key-patient}$ key_1

end enrich

The Implementation

Entity-preEntity-Patient =
module
  export Patient
  refinement
    representation of sorts

pre-patient	implements	patient;
p-keysort-patient	implements	keysort-patient;

    representation of operations

error-patient#	implements	error-patient;
create-patient#	implements	create-patient;
patient_id#	implements	patient_id;
name#	implements	name;
sex#	implements	sex;
birthdate#	implements	birthdate;
birthplace#	implements	birthplace;
costebearer#	implements	costebearer;
address#	implements	address;
famdoctor#	implements	famdoctor;
physicaldata#	implements	physicaldata;
ward#	implements	ward;
room#	implements	room;
set-patient_id#	implements	set-patient_id;
set-name#	implements	set-name;
set-sex#	implements	set-sex;
set-birthdate#	implements	set-birthdate;
set-birthplace#	implements	set-birthplace;
set-costebearer#	implements	set-costebearer;
set-address#	implements	set-address;
set-famdoctor#	implements	set-famdoctor;
set-physicaldata#	implements	set-physicaldata;
set-ward#	implements	set-ward;
set-room#	implements	set-room;
key-patient#	implements	key-patient;
mkkey-patient#	implements	mkkey-patient;

key-patient≪# implements $\ll_{key-patient}$;

import prePatient

procedures

error-patient#	()	: pre-patient;
create-patient#	(patids, names, tsex, dates, place, bearers, adr, physician, physical-data, wards, rooms)	: pre-patient;
patient_id#	(pre-patient)	: patids;
name#	(pre-patient)	: names;
sex#	(pre-patient)	: tsex;
birthdate#	(pre-patient)	: dates;
birthplace#	(pre-patient)	: place;
costebearer#	(pre-patient)	: bearers;
address#	(pre-patient)	: adr;
famdoctor#	(pre-patient)	: physician;
physicaldata#	(pre-patient)	: physical-data;
ward#	(pre-patient)	: wards;
room#	(pre-patient)	: rooms;
set-patient_id#	(pre-patient, patids)	: pre-patient;
set-name#	(pre-patient, names)	: pre-patient;
set-sex#	(pre-patient, tsex)	: pre-patient;
set-birthdate#	(pre-patient, dates)	: pre-patient;
set-birthplace#	(pre-patient, place)	: pre-patient;
set-costebearer#	(pre-patient, bearers)	: pre-patient;
set-address#	(pre-patient, adr)	: pre-patient;
set-famdoctor#	(pre-patient, physician)	: pre-patient;
set-physicaldata#	(pre-patient, physical-data)	: pre-patient;
set-ward#	(pre-patient, wards)	: pre-patient;
set-room#	(pre-patient, rooms)	: pre-patient;
key-patient#	(pre-patient)	: p-keysort-patient;
mkkey-patient#	(patids)	: p-keysort-patient;
key-patient≪#	(p-keysort-patient, p-keysort-patient)	: bool;

variables

b: bool;
a: patids;
a_0: names;
a_1: tsex;
a_2: dates;
a_3: place;
a_4: bearers;
a_5: adr;
a_6: physician;
a_7: physical-data;
a_8: wards;
a_9: rooms;

implementation

error-patient#(**var** $pent_1$)
begin
 $pent_1$:= p-error-patient
end

create-patient#(a, a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9; **var** $pent_1$)
begin
  if a = undef-patids
    ∨ a_0 = undef-names
    ∨ a_1 = undef-tsex
    ∨ a_2 = undef-dates
    ∨ a_3 = undef-place
    ∨ a_4 = undef-bearers
    ∨ a = error-patids
    ∨ a_0 = error-names
    ∨ a_1 = error-tsex
    ∨ a_2 = error-dates
    ∨ a_3 = error-place
    ∨ a_4 = error-bearers
    ∨ a_5 = error-adr
    ∨ a_6 = error-physician
    ∨ a_7 = error-physical-data
    ∨ a_8 = error-wards
    ∨ a_9 = error-rooms **then**
   $pent_1$:= p-error-patient
  else
   $pent_1$:= p-mk-patient(a, a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9)
end

patient_id#($pent_1$; **var** a)
begin
  if $pent_1$ = p-error-patient **then** a := error-patids **else**
   a := p-patient_id($pent_1$)
end

name#($pent_1$; **var** a_0)
begin
  if $pent_1$ = p-error-patient **then** a_0 := error-names **else**
   a_0 := p-name($pent_1$)
end

sex#($pent_1$; **var** a_1)
begin
  if $pent_1$ = p-error-patient **then** a_1 := error-tsex **else**
   a_1 := p-sex($pent_1$)
end

birthdate#($pent_1$; **var** a_2)
begin
  if $pent_1$ = p-error-patient **then** a_2 := error-dates **else**
   a_2 := p-birthdate($pent_1$)
end

birthplace#(pent$_1$; **var** a$_3$)
begin
  if pent$_1$ = p-error-patient **then** a$_3$:= error-place **else**
    a$_3$:= p-birthplace(pent$_1$)
end

costebearer#(pent$_1$; **var** a$_4$)
begin
  if pent$_1$ = p-error-patient **then** a$_4$:= error-bearers **else**
    a$_4$:= p-costebearer(pent$_1$)
end

address#(pent$_1$; **var** a$_5$)
begin
  if pent$_1$ = p-error-patient **then** a$_5$:= error-adr **else**
    a$_5$:= p-address(pent$_1$)
end

famdoctor#(pent$_1$; **var** a$_6$)
begin
  if pent$_1$ = p-error-patient **then** a$_6$:= error-physician **else**
    a$_6$:= p-famdoctor(pent$_1$)
end

physicaldata#(pent$_1$; **var** a$_7$)
begin
  if pent$_1$ = p-error-patient **then** a$_7$:= error-physical-data **else**
    a$_7$:= p-physicaldata(pent$_1$)
end

ward#(pent$_1$; **var** a$_8$)
begin
  if pent$_1$ = p-error-patient **then** a$_8$:= error-wards **else**
    a$_8$:= p-ward(pent$_1$)
end

room#(pent$_1$; **var** a$_9$)
begin
  if pent$_1$ = p-error-patient **then** a$_9$:= error-rooms **else**
    a$_9$:= p-room(pent$_1$)
end

```
set-patient_id#(pent_1, a; var pent1-_1)
begin
  if pent_1 = p-error-patient ∨ a = undef-patids ∨ a = error-patids then
    pent1-_1 := p-error-patient
  else
    pent1-_1 := p-mk-patient(a,
                             p-name(pent_1),
                             p-sex(pent_1),
                             p-birthdate(pent_1),
                             p-birthplace(pent_1),
                             p-costebearer(pent_1),
                             p-address(pent_1),
                             p-famdoctor(pent_1),
                             p-physicaldata(pent_1),
                             p-ward(pent_1),
                             p-room(pent_1))
end
```

```
set-name#(pent_1, a_0; var pent1-_1)
begin
  if pent_1 = p-error-patient ∨ a_0 = undef-names ∨ a_0 = error-names then
    pent1-_1 := p-error-patient
  else
    pent1-_1 := p-mk-patient(p-patient_id(pent_1),
                             a_0,
                             p-sex(pent_1),
                             p-birthdate(pent_1),
                             p-birthplace(pent_1),
                             p-costebearer(pent_1),
                             p-address(pent_1),
                             p-famdoctor(pent_1),
                             p-physicaldata(pent_1),
                             p-ward(pent_1),
                             p-room(pent_1))
end
```

```
set-sex#(pent_1, a_1; var pent1-_1)
begin
  if pent_1 = p-error-patient ∨ a_1 = undef-tsex ∨ a_1 = error-tsex then
    pent1-_1 := p-error-patient
  else
    pent1-_1 := p-mk-patient(p-patient_id(pent_1),
                             p-name(pent_1),
                             a_1,
                             p-birthdate(pent_1),
                             p-birthplace(pent_1),
                             p-costebearer(pent_1),
                             p-address(pent_1),
                             p-famdoctor(pent_1),
                             p-physicaldata(pent_1),
                             p-ward(pent_1),
                             p-room(pent_1))
end
```

```
set-birthdate#(pent1, a2; var pent1-1)
begin
  if pent1 = p-error-patient ∨ a2 = undef-dates ∨ a2 = error-dates then
    pent1-1 := p-error-patient
  else
    pent1-1 := p-mk-patient(p-patient_id(pent1),
                            p-name(pent1),
                            p-sex(pent1),
                            a2,
                            p-birthplace(pent1),
                            p-costebearer(pent1),
                            p-address(pent1),
                            p-famdoctor(pent1),
                            p-physicaldata(pent1),
                            p-ward(pent1),
                            p-room(pent1))
end
```

```
set-birthplace#(pent1, a3; var pent1-1)
begin
  if pent1 = p-error-patient ∨ a3 = undef-place ∨ a3 = error-place then
    pent1-1 := p-error-patient
  else
    pent1-1 := p-mk-patient(p-patient_id(pent1),
                            p-name(pent1),
                            p-sex(pent1),
                            p-birthdate(pent1),
                            a3,
                            p-costebearer(pent1),
                            p-address(pent1),
                            p-famdoctor(pent1),
                            p-physicaldata(pent1),
                            p-ward(pent1),
                            p-room(pent1))
end
```

```
set-costebearer#(pent1, a4; var pent1-1)
begin
  if pent1 = p-error-patient ∨ a4 = undef-bearers ∨ a4 = error-bearers then
    pent1-1 := p-error-patient
  else
    pent1-1 := p-mk-patient(p-patient_id(pent1),
                            p-name(pent1),
                            p-sex(pent1),
                            p-birthdate(pent1),
                            p-birthplace(pent1),
                            a4,
                            p-address(pent1),
                            p-famdoctor(pent1),
                            p-physicaldata(pent1),
                            p-ward(pent1),
                            p-room(pent1))
end
```

```
set-address#(pent_1, a_5; var pent1-_1)
begin
  if pent_1 = p-error-patient ∨ a_5 = error-adr then
    pent1-_1 := p-error-patient
  else
    pent1-_1 := p-mk-patient(p-patient_id(pent_1),
                             p-name(pent_1),
                             p-sex(pent_1),
                             p-birthdate(pent_1),
                             p-birthplace(pent_1),
                             p-costebearer(pent_1),
                             a_5,
                             p-famdoctor(pent_1),
                             p-physicaldata(pent_1),
                             p-ward(pent_1),
                             p-room(pent_1))
end
```

```
set-famdoctor#(pent_1, a_6; var pent1-_1)
begin
  if pent_1 = p-error-patient ∨ a_6 = error-physician then
    pent1-_1 := p-error-patient
  else
    pent1-_1 := p-mk-patient(p-patient_id(pent_1),
                             p-name(pent_1),
                             p-sex(pent_1),
                             p-birthdate(pent_1),
                             p-birthplace(pent_1),
                             p-costebearer(pent_1),
                             p-address(pent_1),
                             a_6,
                             p-physicaldata(pent_1),
                             p-ward(pent_1),
                             p-room(pent_1))
end
```

```
set-physicaldata#(pent_1, a_7; var pent1-_1)
begin
  if pent_1 = p-error-patient ∨ a_7 = error-physical-data then
    pent1-_1 := p-error-patient
  else
    pent1-_1 := p-mk-patient(p-patient_id(pent_1),
                             p-name(pent_1),
                             p-sex(pent_1),
                             p-birthdate(pent_1),
                             p-birthplace(pent_1),
                             p-costebearer(pent_1),
                             p-address(pent_1),
                             p-famdoctor(pent_1),
                             a_7,
                             p-ward(pent_1),
                             p-room(pent_1))
end
```

```
set-ward#(pent₁, a₈; var pent1-₁)
begin
   if pent₁ = p-error-patient ∨ a₈ = error-wards then
     pent1-₁ := p-error-patient
   else
     pent1-₁ := p-mk-patient(p-patient_id(pent₁),
                             p-name(pent₁),
                             p-sex(pent₁),
                             p-birthdate(pent₁),
                             p-birthplace(pent₁),
                             p-costebearer(pent₁),
                             p-address(pent₁),
                             p-famdoctor(pent₁),
                             p-physicaldata(pent₁),
                             a₈,
                             p-room(pent₁))
end
```

```
set-room#(pent₁, a₉; var pent1-₁)
begin
   if pent₁ = p-error-patient ∨ a₉ = error-rooms then
     pent1-₁ := p-error-patient
   else
     pent1-₁ := p-mk-patient(p-patient_id(pent₁),
                             p-name(pent₁),
                             p-sex(pent₁),
                             p-birthdate(pent₁),
                             p-birthplace(pent₁),
                             p-costebearer(pent₁),
                             p-address(pent₁),
                             p-famdoctor(pent₁),
                             p-physicaldata(pent₁),
                             p-ward(pent₁),
                             a₉)
end
```

```
key-patient#(pent₁; var pkey₁)
begin
   if pent₁ = p-error-patient then pkey₁ := p-mkkey-patient(error-patient_id) else
     pkey₁ := p-mkkey-patient(p-patient_id(pent₁))
end
```

```
mkkey-patient#(a; var pkey₁)
begin
   pkey₁ := p-mkkey-patient(a)
end
```

```
key-patient≪#(pkey₁, pkey1-₁; var b)
begin
   if k-patient_id(pkey₁) ≪_patids k-patient_id(pkey1-₁) then b := tt else b := ff
end
```

```
rs-patient#(pent_1)
begin
  if pent_1 = p-error-patient then skip else
    var pent1-_1 = p-error-patient in
    begin
      create-patient#(p-patient_id(pent_1),
                      p-name(pent_1),
                      p-sex(pent_1),
                      p-birthdate(pent_1),
                      p-birthplace(pent_1),
                      p-costebearer(pent_1),
                      p-address(pent_1),
                      p-famdoctor(pent_1),
                      p-physicaldata(pent_1),
                      p-ward(pent_1),
                      p-room(pent_1);
                      pent1-_1);
      if pent1-_1 = p-error-patient then abort
    end
end
```

```
rs-key-patient#(pkey_1)
begin skip end
```

A.2.2 CC_OR

The Specifications

The Specification preCC_OR An instantiation of the scheme preEntity_i.

```
 preCC_OR =
data specification
    using OrderedAttributes
    pre-cc_or =  p-mk-cc_or (p-ccor_id : ccid,
                             p-makingdate : dates,
                             p-comment : text)
               | p-error-cc_or
               ;
    p-keysort-cc_or = p-mkkey-cc_or (k-ccor_id : ccid);
    variables
        pent3-_1, pent_3: pre-cc_or;
        pkey3-_1, pkey_3: p-keysort-cc_or;
end data specification
```

The Specification CC_OR An instantiation of the scheme Entity_i.

```
 CC_OR =
enrich OrderedAttributes with
    sorts cc_or, keysort-cc_or;
    constants error-cc_or : cc_or;
    functions
        create-cc_or  :  ccid × dates × text  →  cc_or  ;
        ccor_id       :  cc_or                →  ccid   ;
        makingdate    :  cc_or                →  dates  ;
        comment       :  cc_or                →  text   ;
```

set-ccor_id	:	cc_or × ccid	→	cc_or	;
set-makingdate	:	cc_or × dates	→	cc_or	;
set-comment	:	cc_or × text	→	cc_or	;
key-cc_or	:	cc_or	→	keysort-cc_or	;
mkkey-cc_or	:	ccid	→	keysort-cc_or	;

predicates $. \ll_{key-cc\_or} .$: keysort-cc_or × keysort-cc_or;

variables

ent_3: cc_or;
key3-$_2$, key3-$_1$, key$_3$: keysort-cc_or;
a_{40}, a_{37}: ccid;
a_{41}, a_{38}: dates;
a_{42}, a_{39}: text;

axioms

cc_or **generated by** create-cc_or, error-cc_or;
keysort-cc_or **freely generated by** mkkey-cc_or;
a_{37} = undef-ccid
∨ a_{38} = undef-dates
∨ a_{37} = error-ccid
∨ a_{38} = error-dates
∨ a_{39} = error-text
↔
create-cc_or(a_{37}, a_{38}, a_{39}) = error-cc_or,
create-cc_or(a_{37}, a_{38}, a_{39}) ≠ error-cc_or
→ (create-cc_or(a_{37}, a_{38}, a_{39})= create-cc_or(a_{40}, a_{41}, a_{42})
  → a_{37} = a_{40}
   ∧ a_{38} = a_{41}
   ∧ a_{39} = a_{42})
 ∧ ccor_id(create-cc_or(a_{37}, a_{38}, a_{39})) = a_{37}
 ∧ makingdate(create-cc_or(a_{37}, a_{38}, a_{39})) = a_{38}
 ∧ comment(create-cc_or(a_{37}, a_{38}, a_{39})) = a_{39},
ent_3 ≠ error-cc_or
→ set-ccor_id(ent_3, a_{37})
 = create-cc_or(a_{37},
    makingdate(ent_3),
    comment(ent_3))
 ∧ set-makingdate(ent_3, a_{38})
  = create-cc_or(ccor_id(ent_3),
    a_{38},
    comment(ent_3))
 ∧ set-comment(ent_3, a_{39})
  = create-cc_or(ccor_id(ent_3),
    makingdate(ent_3),
    a_{39}),
ccor_id(error-cc_or) = error-ccid,
makingdate(error-cc_or) = error-dates,
comment(error-cc_or) = error-text,
set-ccor_id(error-cc_or, a_{37}) = error-cc_or,
set-makingdate(error-cc_or, a_{38}) = error-cc_or,
set-comment(error-cc_or, a_{39}) = error-cc_or,
key-cc_or(ent_3) = mkkey-cc_or(ccor_id(ent_3)),
mkkey-cc_or(a_{37}) $\ll_{key-cc\_or}$ mkkey-cc_or(a_{40}) ↔ $a_{37} \ll_{ccid} a_{40}$,
¬ key$_3$ $\ll_{key-cc\_or}$ key$_3$,
key$_3$ $\ll_{key-cc\_or}$ key3-$_1$ ∧ key3-$_1$ $\ll_{key-cc\_or}$ key3-$_2$ → key$_3$ $\ll_{key-cc\_or}$ key3-$_2$,
key$_3$ $\ll_{key-cc\_or}$ key3-$_1$ ∨ key$_3$ = key3-$_1$ ∨ key3-$_1$ $\ll_{key-cc\_or}$ key$_3$

end enrich

The Implementation

```
 Entity-preEntity-CC_OR =
module
    export CC_OR
    refinement
      representation of sorts
          pre-cc_or          implements   cc_or;
          p-keysort-cc_or    implements   keysort-cc_or;
      representation of operations
          error-cc_or#        implements   error-cc_or;
          create-cc_or#       implements   create-cc_or;
          ccor_id#            implements   ccor_id;
          makingdate#         implements   makingdate;
          comment#            implements   comment;
          set-ccor_id#        implements   set-ccor_id;
          set-makingdate#     implements   set-makingdate;
          set-comment#        implements   set-comment;
          key-cc_or#          implements   key-cc_or;
          mkkey-cc_or#        implements   mkkey-cc_or;
          key-cc_or≪#         implements   ≪_{key-cc_or};

      import preCC_OR

      procedures
           error-cc_or#      ()                                  : pre-cc_or;
           create-cc_or#     (ccid, dates, text)                 : pre-cc_or;
           ccor_id#          (pre-cc_or)                         : ccid;
           makingdate#       (pre-cc_or)                         : dates;
           comment#          (pre-cc_or)                         : text;
           set-ccor_id#      (pre-cc_or, ccid)                   : pre-cc_or;
           set-makingdate#   (pre-cc_or, dates)                  : pre-cc_or;
           set-comment#      (pre-cc_or, text)                   : pre-cc_or;
           key-cc_or#        (pre-cc_or)                         : p-keysort-cc_or;
           mkkey-cc_or#      (ccid)                              : p-keysort-cc_or;
           key-cc_or≪#       (p-keysort-cc_or, p-keysort-cc_or)  : bool;

      variables b: bool; a37: ccid; a38: dates; a39: text;

      implementation


error-cc_or#(var pent3)
begin
  pent3 := p-error-cc_or
end


create-cc_or#(a37, a38, a39; var pent3)
begin
  if a37 = undef-ccid
     ∨ a38 = undef-dates
     ∨ a37 = error-ccid
     ∨ a38 = error-dates
     ∨ a39 = error-text then
    pent3 := p-error-cc_or
```

else
 $pent_3$:= p-mk-cc_or(a_{37}, a_{38}, a_{39})
end

ccor_id#($pent_3$; **var** a_{37})
begin
 if $pent_3$ = p-error-cc_or **then** a_{37} := error-ccid **else** a_{37} := p-ccor_id($pent_3$)
end

makingdate#($pent_3$; **var** a_{38})
begin
 if $pent_3$ = p-error-cc_or **then** a_{38} := error-dates **else** a_{38} := p-makingdate($pent_3$)
end

comment#($pent_3$; **var** a_{39})
begin
 if $pent_3$ = p-error-cc_or **then** a_{39} := error-text **else** a_{39} := p-comment($pent_3$)
end

set-ccor_id#($pent_3$, a_{37}; **var** pent3-$_1$)
begin
 if $pent_3$ = p-error-cc_or $\vee$ a_{37} = undef-ccid $\vee$ a_{37} = error-ccid **then**
 pent3-$_1$:= p-error-cc_or
 else
 pent3-$_1$:= p-mk-cc_or(a_{37}, p-makingdate($pent_3$), p-comment($pent_3$))
end

set-makingdate#($pent_3$, a_{38}; **var** pent3-$_1$)
begin
 if $pent_3$ = p-error-cc_or $\vee$ a_{38} = undef-dates $\vee$ a_{38} = error-dates **then**
 pent3-$_1$:= p-error-cc_or
 else
 pent3-$_1$:= p-mk-cc_or(p-ccor_id($pent_3$), a_{38}, p-comment($pent_3$))
end

set-comment#($pent_3$, a_{39}; **var** pent3-$_1$)
begin
 if $pent_3$ = p-error-cc_or $\vee$ a_{39} = error-text **then** pent3-$_1$:= p-error-cc_or **else**
 pent3-$_1$:= p-mk-cc_or(p-ccor_id($pent_3$), p-makingdate($pent_3$), a_{39})
end

key-cc_or#($pent_3$; **var** $pkey_3$)
begin
 if $pent_3$ = p-error-cc_or **then** $pkey_3$:= p-mkkey-cc_or(error-ccor_id) **else**
 $pkey_3$:= p-mkkey-cc_or(p-ccor_id($pent_3$))
end

```
mkkey-cc_or#(a37; var pkey3)
begin
  pkey3 := p-mkkey-cc_or(a37)
end
```

```
key-cc_or≪#(pkey3, pkey3-1; var b)
begin
  if k-ccor_id(pkey3) ≪ccid k-ccor_id(pkey3-1) then b := tt else b := ff
end
```

```
rs-cc_or#(pent3)
begin
  if pent3 = p-error-cc_or then skip else
    var pent3-1 = p-error-cc_or in
    begin
      create-cc_or#(p-ccor_id(pent3),
                    p-makingdate(pent3),
                    p-comment(pent3);
                    pent3-1);
      if pent3-1 = p-error-cc_or then abort
    end
end
```

```
rs-key-cc_or#(pkey3)
begin skip end
```

A.2.3 CC_Data

The Specifications

The Specification preCC_Data An instantiation of the scheme preEntity$_i$.

```
preCC_Data =
data specification
  using OrderedAttributes
  pre-cc_data = p-mk-cc_data (p-ccor_id2 : ccid,
                              p-ccr : text,
                              p-examinationdate : dates,
                              p-start : time,
                              p-end : time,
                              p-pressurecurves : curves,
                              p-x-ray-film : film)
              | p-error-cc_data
              ;
  p-keysort-cc_data = p-mkkey-cc_data (k-ccor_id2 : ccid);
  variables
      pent4-1, pent4: pre-cc_data;
      pkey4-1, pkey4: p-keysort-cc_data;
end data specification
```

The Specification CC_Data An instantiation of the scheme Entity_i.

CC_Data =
enrich OrderedAttributes **with**
sorts cc_data, keysort-cc_data;
constants error-cc_data : cc_data;
functions

create-cc_data	:	ccid × text × dates × time			
		× time × curves × film	→	cc_data	;
ccor_id_2	:	cc_data	→	ccid	;
ccr	:	cc_data	→	text	;
examinationdate	:	cc_data	→	dates	;
start	:	cc_data	→	time	;
end	:	cc_data	→	time	;
pressurecurves	:	cc_data	→	curves	;
x-ray-film	:	cc_data	→	film	;
set-ccor_id_2	:	cc_data × ccid	→	cc_data	;
set-ccr	:	cc_data × text	→	cc_data	;
set-examinationdate	:	cc_data × dates	→	cc_data	;
set-start	:	cc_data × time	→	cc_data	;
set-end	:	cc_data × time	→	cc_data	;
set-pressurecurves	:	cc_data × curves	→	cc_data	;
set-x-ray-film	:	cc_data × film	→	cc_data	;
key-cc_data	:	cc_data	→	keysort-cc_data	;
mkkey-cc_data	:	ccid	→	keysort-cc_data	;

predicates $. \ll_{key-cc\_data} .$: keysort-cc_data × keysort-cc_data;
variables
ent_4: cc_data;
key4-_2, key4-_1, key_4: keysort-cc_data;
a_{50}, a_{43}: ccid;
a_{51}, a_{44}: text;
a_{52}, a_{45}: dates;
a_{54}, a_{53}, a_{47}, a_{46}: time;
a_{55}, a_{48}: curves;
a_{56}, a_{49}: film;
axioms
cc_data **generated by** create-cc_data, error-cc_data;
keysort-cc_data **freely generated by** mkkey-cc_data;
$a_{43} = \text{undef-ccid}$
$\vee\ a_{45} = \text{undef-dates}$
$\vee\ a_{46} = \text{undef-time}$
$\vee\ a_{43} = \text{error-ccid}$
$\vee\ a_{44} = \text{error-text}$
$\vee\ a_{45} = \text{error-dates}$
$\vee\ a_{46} = \text{error-time}$
$\vee\ a_{47} = \text{error-time}$
$\vee\ a_{48} = \text{error-curves}$
$\vee\ a_{49} = \text{error-film}$
$\leftrightarrow$
$\text{create-cc\_data}(a_{43}, a_{44}, a_{45}, a_{46}, a_{47}, a_{48}, a_{49}) = \text{error-cc\_data}$,
$\text{create-cc\_data}(a_{43}, a_{44}, a_{45}, a_{46}, a_{47}, a_{48}, a_{49}) \neq \text{error-cc\_data}$
$\rightarrow (\text{create-cc\_data}(a_{43}, a_{44}, a_{45}, a_{46}, a_{47}, a_{48}, a_{49})$
$= \text{create-cc\_data}(a_{50}, a_{51}, a_{52}, a_{53}, a_{54}, a_{55}, a_{56})$
$\rightarrow a_{43} = a_{50}$
$\wedge\ a_{44} = a_{51}$
$\wedge\ a_{45} = a_{52}$
$\wedge\ a_{46} = a_{53}$

$\wedge$ $a_{47} = a_{54}$
$\wedge$ $a_{48} = a_{55}$
$\wedge$ $a_{49} = a_{56}$)
$\wedge$ ccor_id$_2$(create-cc_data(a_{43}, a_{44}, a_{45}, a_{46}, a_{47}, a_{48}, a_{49})) = a_{43}
$\wedge$ ccr(create-cc_data(a_{43}, a_{44}, a_{45}, a_{46}, a_{47}, a_{48}, a_{49})) = a_{44}
$\wedge$ examinationdate(create-cc_data(a_{43}, a_{44}, a_{45}, a_{46}, a_{47}, a_{48}, a_{49})) = a_{45}
$\wedge$ start(create-cc_data(a_{43}, a_{44}, a_{45}, a_{46}, a_{47}, a_{48}, a_{49})) = a_{46}
$\wedge$ end(create-cc_data(a_{43}, a_{44}, a_{45}, a_{46}, a_{47}, a_{48}, a_{49})) = a_{47}
$\wedge$ pressurecurves(create-cc_data(a_{43}, a_{44}, a_{45}, a_{46}, a_{47}, a_{48}, a_{49})) = a_{48}
$\wedge$ x-ray-film(create-cc_data(a_{43}, a_{44}, a_{45}, a_{46}, a_{47}, a_{48}, a_{49})) = a_{49},
ent$_4$ $\neq$ error-cc_data
$\rightarrow$ set-ccor_id$_2$(ent$_4$, a_{43})
= create-cc_data(a_{43},
ccr(ent$_4$),
examinationdate(ent$_4$),
start(ent$_4$),
end(ent$_4$),
pressurecurves(ent$_4$),
x-ray-film(ent$_4$))
$\wedge$ set-ccr(ent$_4$, a_{44})
= create-cc_data(ccor_id$_2$(ent$_4$),
a_{44},
examinationdate(ent$_4$),
start(ent$_4$),
end(ent$_4$),
pressurecurves(ent$_4$),
x-ray-film(ent$_4$))
$\wedge$ set-examinationdate(ent$_4$, a_{45})
= create-cc_data(ccor_id$_2$(ent$_4$),
ccr(ent$_4$),
a_{45},
start(ent$_4$),
end(ent$_4$),
pressurecurves(ent$_4$),
x-ray-film(ent$_4$))
$\wedge$ set-start(ent$_4$, a_{46})
= create-cc_data(ccor_id$_2$(ent$_4$),
ccr(ent$_4$),
examinationdate(ent$_4$),
a_{46},
end(ent$_4$),
pressurecurves(ent$_4$),
x-ray-film(ent$_4$))
$\wedge$ set-end(ent$_4$, a_{47})
= create-cc_data(ccor_id$_2$(ent$_4$),
ccr(ent$_4$),
examinationdate(ent$_4$),
start(ent$_4$),
a_{47},
pressurecurves(ent$_4$),
x-ray-film(ent$_4$))
$\wedge$ set-pressurecurves(ent$_4$, a_{48})
= create-cc_data(ccor_id$_2$(ent$_4$),
ccr(ent$_4$),
examinationdate(ent$_4$),
start(ent$_4$),

end(ent_4),
a_{48},
x-ray-film(ent_4))
$\land$ set-x-ray-film(ent_4, a_{49})
= create-cc_data(ccor_id$_2$(ent_4),
ccr(ent_4),
examinationdate(ent_4),
start(ent_4),
end(ent_4),
pressurecurves(ent_4),
a_{49}),
ccor_id$_2$(error-cc_data) = error-ccid,
ccr(error-cc_data) = error-text,
examinationdate(error-cc_data) = error-dates,
start(error-cc_data) = error-time,
end(error-cc_data) = error-time,
pressurecurves(error-cc_data) = error-curves,
x-ray-film(error-cc_data) = error-film,
set-ccor_id$_2$(error-cc_data, a_{43}) = error-cc_data,
set-ccr(error-cc_data, a_{44}) = error-cc_data,
set-examinationdate(error-cc_data, a_{45}) = error-cc_data,
set-start(error-cc_data, a_{46}) = error-cc_data,
set-end(error-cc_data, a_{47}) = error-cc_data,
set-pressurecurves(error-cc_data, a_{48}) = error-cc_data,
set-x-ray-film(error-cc_data, a_{49}) = error-cc_data,
key-cc_data(ent_4) = mkkey-cc_data(ccor_id$_2$(ent_4)),
mkkey-cc_data(a_{43}) $\ll_{key-cc\_data}$ mkkey-cc_data(a_{50})
$\leftrightarrow a_{43} \ll_{ccid} a_{50}$,
$\neg\ key_4 \ll_{key-cc\_data} key_4$,
$key_4 \ll_{key-cc\_data} key4\text{-}_1 \land key4\text{-}_1 \ll_{key-cc\_data} key4\text{-}_2$
$\rightarrow key_4 \ll_{key-cc\_data} key4\text{-}_2$,
$key_4 \ll_{key-cc\_data} key4\text{-}_1 \lor key_4 = key4\text{-}_1 \lor key4\text{-}_1 \ll_{key-cc\_data} key_4$
end enrich

The Implementation

Entity-preEntity-CC_Data =
module
export CC_Data
refinement
representation of sorts

pre-cc_data	implements	cc_data;
p-keysort-cc_data	implements	keysort-cc_data;

representation of operations

error-cc_data#	implements	error-cc_data;
create-cc_data#	implements	create-cc_data;
ccor_id2#	implements	ccor_id$_2$;
ccr#	implements	ccr;
examinationdate#	implements	examinationdate;
start#	implements	start;
end#	implements	end;
pressurecurves#	implements	pressurecurves;
x-ray-film#	implements	x-ray-film;
set-ccor_id2#	implements	set-ccor_id$_2$;
set-ccr#	implements	set-ccr;
set-examinationdate#	implements	set-examinationdate;

```
        set-start#             implements   set-start;
        set-end#               implements   set-end;
        set-pressurecurves#    implements   set-pressurecurves;
        set-x-ray-film#        implements   set-x-ray-film;
        key-cc_data#           implements   key-cc_data;
        mkkey-cc_data#         implements   mkkey-cc_data;
        key-cc_data≪#          implements   ≪key-cc_data;

    import preCC_Data

    procedures
        error-cc_data#         ()                          : pre-cc_data;
        create-cc_data#        (ccid, text, dates, time,
                               time, curves, film)         : pre-cc_data;
        ccor_id2#              (pre-cc_data)               : ccid;
        ccr#                   (pre-cc_data)               : text;
        examinationdate#       (pre-cc_data)               : dates;
        start#                 (pre-cc_data)               : time;
        end#                   (pre-cc_data)               : time;
        pressurecurves#        (pre-cc_data)               : curves;
        x-ray-film#            (pre-cc_data)               : film;
        set-ccor_id2#          (pre-cc_data, ccid)         : pre-cc_data;
        set-ccr#               (pre-cc_data, text)         : pre-cc_data;
        set-examinationdate#   (pre-cc_data, dates)        : pre-cc_data;
        set-start#             (pre-cc_data, time)         : pre-cc_data;
        set-end#               (pre-cc_data, time)         : pre-cc_data;
        set-pressurecurves#    (pre-cc_data, curves)       : pre-cc_data;
        set-x-ray-film#        (pre-cc_data, film)         : pre-cc_data;
        key-cc_data#           (pre-cc_data)               : p-keysort-cc_data;
        mkkey-cc_data#         (ccid)                      : p-keysort-cc_data;
        key-cc_data≪#          (p-keysort-cc_data,
                               p-keysort-cc_data)          : bool;

    variables
        b: bool;
        a43: ccid;
        a44: text;
        a45: dates;
        a47, a46: time;
        a48: curves;
        a49: film;

    implementation


error-cc_data#(var pent4)
begin
  pent4 := p-error-cc_data
end


create-cc_data#(a43, a44, a45, a46, a47, a48, a49; var pent4)
begin
  if a43 = undef-ccid
    ∨ a45 = undef-dates
```

```
      ∨ a_46 = undef-time
      ∨ a_43 = error-ccid
      ∨ a_44 = error-text
      ∨ a_45 = error-dates
      ∨ a_46 = error-time
      ∨ a_47 = error-time
      ∨ a_48 = error-curves
      ∨ a_49 = error-film then
     pent_4 := p-error-cc_data
   else
     pent_4 := p-mk-cc_data(a_43, a_44, a_45, a_46, a_47, a_48, a_49)
end
```

```
ccor_id2#(pent_4; var a_43)
begin
   if pent_4 = p-error-cc_data then a_43 := error-ccid else
     a_43 := p-ccor_id_2(pent_4)
end
```

```
ccr#(pent_4; var a_44)
begin
   if pent_4 = p-error-cc_data then a_44 := error-text else
     a_44 := p-ccr(pent_4)
end
```

```
examinationdate#(pent_4; var a_45)
begin
   if pent_4 = p-error-cc_data then a_45 := error-dates else
     a_45 := p-examinationdate(pent_4)
end
```

```
start#(pent_4; var a_46)
begin
   if pent_4 = p-error-cc_data then a_46 := error-time else
     a_46 := p-start(pent_4)
end
```

```
end#(pent_4; var a_47)
begin
   if pent_4 = p-error-cc_data then a_47 := error-time else
     a_47 := p-end(pent_4)
end
```

```
pressurecurves#(pent_4; var a_48)
begin
   if pent_4 = p-error-cc_data then a_48 := error-curves else
     a_48 := p-pressurecurves(pent_4)
end
```

```
x-ray-film#(pent4; var a49)
begin
  if pent4 = p-error-cc_data then a49 := error-film else
    a49 := p-x-ray-film(pent4)
end
```

```
set-ccor_id2#(pent4, a43; var pent4-1)
begin
  if pent4 = p-error-cc_data ∨ a43 = undef-ccid ∨ a43 = error-ccid then
    pent4-1 := p-error-cc_data
  else
    pent4-1 := p-mk-cc_data(a43,
                            p-ccr(pent4),
                            p-examinationdate(pent4),
                            p-start(pent4),
                            p-end(pent4),
                            p-pressurecurves(pent4),
                            p-x-ray-film(pent4))
  end
```

```
set-ccr#(pent4, a44; var pent4-1)
begin
  if pent4 = p-error-cc_data ∨ a44 = error-text then
    pent4-1 := p-error-cc_data
  else
    pent4-1 := p-mk-cc_data(p-ccor_id2(pent4),
                            a44,
                            p-examinationdate(pent4),
                            p-start(pent4),
                            p-end(pent4),
                            p-pressurecurves(pent4),
                            p-x-ray-film(pent4))
end
```

```
set-examinationdate#(pent4, a45; var pent4-1)
begin
  if pent4 = p-error-cc_data ∨ a45 = undef-dates ∨ a45 = error-dates then
    pent4-1 := p-error-cc_data
  else
    pent4-1 := p-mk-cc_data(p-ccor_id2(pent4),
                            p-ccr(pent4),
                            a45,
                            p-start(pent4),
                            p-end(pent4),
                            p-pressurecurves(pent4),
                            p-x-ray-film(pent4))
end
```

```
set-start#(pent_4, a_46; var pent4-_1)
begin
  if pent_4 = p-error-cc_data ∨ a_46 = undef-time ∨ a_46 = error-time then
    pent4-_1 := p-error-cc_data
  else
    pent4-_1 := p-mk-cc_data(p-ccor_id_2(pent_4),
                             p-ccr(pent_4),
                             p-examinationdate(pent_4),
                             a_46,
                             p-end(pent_4),
                             p-pressurecurves(pent_4),
                             p-x-ray-film(pent_4))
end


set-end#(pent_4, a_47; var pent4-_1)
begin
  if pent_4 = p-error-cc_data ∨ a_47 = error-time then
    pent4-_1 := p-error-cc_data
  else
    pent4-_1 := p-mk-cc_data(p-ccor_id_2(pent_4),
                             p-ccr(pent_4),
                             p-examinationdate(pent_4),
                             p-start(pent_4),
                             a_47,
                             p-pressurecurves(pent_4),
                             p-x-ray-film(pent_4))
end


set-pressurecurves#(pent_4, a_48; var pent4-_1)
begin
  if pent_4 = p-error-cc_data ∨ a_48 = error-curves then
    pent4-_1 := p-error-cc_data
  else
    pent4-_1 := p-mk-cc_data(p-ccor_id_2(pent_4),
                             p-ccr(pent_4),
                             p-examinationdate(pent_4),
                             p-start(pent_4),
                             p-end(pent_4),
                             a_48,
                             p-x-ray-film(pent_4))
end


set-x-ray-film#(pent_4, a_49; var pent4-_1)
begin
  if pent_4 = p-error-cc_data ∨ a_49 = error-film then
    pent4-_1 := p-error-cc_data
  else
    pent4-_1 := p-mk-cc_data(p-ccor_id_2(pent_4),
                             p-ccr(pent_4),
                             p-examinationdate(pent_4),
                             p-start(pent_4),
                             p-end(pent_4),
```

```
                                   p-pressurecurves(pent4),
                                   a49)
end
```

```
key-cc_data#(pent4; var pkey4)
begin
  if pent4 = p-error-cc_data then pkey4 := p-mkkey-cc_data(error-ccor_id) else
    pkey4 := p-mkkey-cc_data(p-ccor_id2(pent4))
end
```

```
mkkey-cc_data#(a43; var pkey4)
begin
  pkey4 := p-mkkey-cc_data(a43)
end
```

```
key-cc_data≪#(pkey4, pkey4-1; var b)
begin
  if k-ccor_id2(pkey4) ≪ccid k-ccor_id2(pkey4-1) then b := tt else b := ff
end
```

```
rs-cc_data#(pent4)
begin
  if pent4 = p-error-cc_data then skip else
    var pent4-1 = p-error-cc_data in
    begin
       create-cc_data#(p-ccor_id2(pent4),
                       p-ccr(pent4),
                       p-examinationdate(pent4),
                       p-start(pent4),
                       p-end(pent4),
                       p-pressurecurves(pent4),
                       p-x-ray-film(pent4);
                       pent4-1);
       if pent4-1 = p-error-cc_data then abort
    end
end
```

```
rs-key-cc_data#(pkey4)
begin skip end
```

A.2.4 CC_Findings

The Specifications

The Specification preCC_Findings An instantiation of the scheme preEntity_i.

preCC_Findings =
data specification
  using OrderedAttributes
  pre-cc_findings = p-mk-cc_findings (p-findings_id : findingsids,
    p-findingsdate : dates,
    p-findings : tfindings,
    p-reprort : letter)
  | p-error-cc_findings
  ;
  p-keysort-cc_findings = p-mkkey-cc_findings (k-findings_id : findingsids);
  variables
    pent5-_1, pent_5: pre-cc_findings;
    pkey5-_1, pkey_5: p-keysort-cc_findings;
end data specification

The Specification CC_Findings An instantiation of the scheme Entity_i.

CC_Findings =
enrich OrderedAttributes **with**
  sorts cc_findings, keysort-cc_findings;
  constants error-cc_findings : cc_findings;
  functions

create-cc_findings	:	findingsids × dates × tfindings × letter	→	cc_findings	;
findings_id	:	cc_findings	→	findingsids	;
findingsdate	:	cc_findings	→	dates	;
findings	:	cc_findings	→	tfindings	;
reprort	:	cc_findings	→	letter	;
set-findings_id	:	cc_findings × findingsids	→	cc_findings	;
set-findingsdate	:	cc_findings × dates	→	cc_findings	;
set-findings	:	cc_findings × tfindings	→	cc_findings	;
set-reprort	:	cc_findings × letter	→	cc_findings	;
key-cc_findings	:	cc_findings	→	keysort-cc_findings	;
mkkey-cc_findings	:	findingsids	→	keysort-cc_findings	;

  predicates $. \ll_{key-cc\_findings} .$: keysort-cc_findings × keysort-cc_findings;
  variables
    ent_5: cc_findings;
    key5-_2, key5-_1, key_5: keysort-cc_findings;
    a_{61}, a_{57}: findingsids;
    a_{62}, a_{58}: dates;
    a_{63}, a_{59}: tfindings;
    a_{64}, a_{60}: letter;
  axioms
    cc_findings **generated by** create-cc_findings, error-cc_findings;
    keysort-cc_findings **freely generated by** mkkey-cc_findings;
    a_{57} = undef-findingsids
    $\vee$ a_{58} = undef-dates
    $\vee$ a_{59} = undef-tfindings
    $\vee$ a_{57} = error-findingsids
    $\vee$ a_{58} = error-dates
    $\vee$ a_{59} = error-tfindings
    $\vee$ a_{60} = error-letter
    $\leftrightarrow$ create-cc_findings(a_{57}, a_{58}, a_{59}, a_{60}) = error-cc_findings,
    create-cc_findings(a_{57}, a_{58}, a_{59}, a_{60}) $\neq$ error-cc_findings
    $\rightarrow$ (create-cc_findings(a_{57}, a_{58}, a_{59}, a_{60})
      = create-cc_findings(a_{61}, a_{62}, a_{63}, a_{64})

$\rightarrow a_{57} = a_{61}$
$\wedge\ a_{58} = a_{62}$
$\wedge\ a_{59} = a_{63}$
$\wedge\ a_{60} = a_{64})$
$\wedge$ findings_id(create-cc_findings(a_{57}, a_{58}, a_{59}, a_{60})) = a_{57}
$\wedge$ findingsdate(create-cc_findings(a_{57}, a_{58}, a_{59}, a_{60})) = a_{58}
$\wedge$ findings(create-cc_findings(a_{57}, a_{58}, a_{59}, a_{60})) = a_{59}
$\wedge$ reprort(create-cc_findings(a_{57}, a_{58}, a_{59}, a_{60})) = a_{60},
$ent_5 \neq$ error-cc_findings
$\rightarrow$ set-findings_id(ent_5, a_{57})
= create-cc_findings(a_{57},
findingsdate(ent_5),
findings(ent_5),
reprort(ent_5))
$\wedge$ set-findingsdate(ent_5, a_{58})
= create-cc_findings(findings_id(ent_5),
a_{58},
findings(ent_5),
reprort(ent_5))
$\wedge$ set-findings(ent_5, a_{59})
= create-cc_findings(findings_id(ent_5),
findingsdate(ent_5),
a_{59},
reprort(ent_5))
$\wedge$ set-reprort(ent_5, a_{60})
= create-cc_findings(findings_id(ent_5),
findingsdate(ent_5),
findings(ent_5),
a_{60}),
findings_id(error-cc_findings) = error-findingsids,
findingsdate(error-cc_findings) = error-dates,
findings(error-cc_findings) = error-tfindings,
reprort(error-cc_findings) = error-letter,
set-findings_id(error-cc_findings, a_{57}) = error-cc_findings,
set-findingsdate(error-cc_findings, a_{58}) = error-cc_findings,
set-findings(error-cc_findings, a_{59}) = error-cc_findings,
set-reprort(error-cc_findings, a_{60}) = error-cc_findings,
key-cc_findings(ent_5) = mkkey-cc_findings(findings_id(ent_5)),
mkkey-cc_findings(a_{57}) $\ll_{key-cc\_findings}$ mkkey-cc_findings(a_{61})
$\leftrightarrow a_{57} \ll_{findingsids} a_{61}$,
$\neg\ key_5 \ll_{key-cc\_findings} key_5$,
$key_5 \ll_{key-cc\_findings} key5\text{-}_1 \wedge key5\text{-}_1 \ll_{key-cc\_findings} key5\text{-}_2$
$\rightarrow key_5 \ll_{key-cc\_findings} key5\text{-}_2$,
$key_5 \ll_{key-cc\_findings} key5\text{-}_1 \vee key_5 = key5\text{-}_1 \vee key5\text{-}_1 \ll_{key-cc\_findings} key_5$
end enrich

The Implementation

Entity-preEntity-CC_Findings =
module
export CC_Findings
refinement
representation of sorts

pre-cc_findings	implements	cc_findings;
p-keysort-cc_findings	implements	keysort-cc_findings;

representation of operations

```
        error-cc_findings#      implements   error-cc_findings;
        create-cc_findings#     implements   create-cc_findings;
        findings_id#            implements   findings_id;
        findingsdate#           implements   findingsdate;
        findings#               implements   findings;
        reprort#                implements   reprort;
        set-findings_id#        implements   set-findings_id;
        set-findingsdate#       implements   set-findingsdate;
        set-findings#           implements   set-findings;
        set-reprort#            implements   set-reprort;
        key-cc_findings#        implements   key-cc_findings;
        mkkey-cc_findings#      implements   mkkey-cc_findings;
        key-cc_findings≪#       implements   key-cc_findings≪;

    import preCC_Findings

    procedures
        error-cc_findings#      ()                                : pre-cc_findings;
        create-cc_findings#     (findingsids, dates,
                                 tfindings, letter)               : pre-cc_findings;
        findings_id#            (pre-cc_findings)                 : findingsids;
        findingsdate#           (pre-cc_findings)                 : dates;
        findings#               (pre-cc_findings)                 : tfindings;
        reprort#                (pre-cc_findings)                 : letter;
        set-findings_id#        (pre-cc_findings, findingsids)    : pre-cc_findings;
        set-findingsdate#       (pre-cc_findings, dates)          : pre-cc_findings;
        set-findings#           (pre-cc_findings, tfindings)      : pre-cc_findings;
        set-reprort#            (pre-cc_findings, letter)         : pre-cc_findings;
        key-cc_findings#        (pre-cc_findings)                 : p-keysort-cc_findings;
        mkkey-cc_findings#      (findingsids)                     : p-keysort-cc_findings;
        key-cc_findings≪#       (p-keysort-cc_findings,
                                 p-keysort-cc_findings)           : bool;

    variables b: bool; a57: findingsids; a58: dates; a59: tfindings; a60: letter;

    implementation


error-cc_findings#(var pent5)
begin
  pent5 := p-error-cc_findings
end


create-cc_findings#(a57, a58, a59, a60; var pent5)
begin
  if a57 = undef-findingsids
     ∨ a58 = undef-dates
     ∨ a59 = undef-tfindings
     ∨ a57 = error-findingsids
     ∨ a58 = error-dates
     ∨ a59 = error-tfindings
     ∨ a60 = error-letter then
    pent5 := p-error-cc_findings
  else
    pent5 := p-mk-cc_findings(a57, a58, a59, a60)
end
```

findings_id#(pent$_5$; **var** a$_{57}$)
begin
  if pent$_5$ = p-error-cc_findings **then** a$_{57}$:= error-findingsids **else**
    a$_{57}$:= p-findings_id(pent$_5$)
end

findingsdate#(pent$_5$; **var** a$_{58}$)
begin
  if pent$_5$ = p-error-cc_findings **then** a$_{58}$:= error-dates **else**
    a$_{58}$:= p-findingsdate(pent$_5$)
end

findings#(pent$_5$; **var** a$_{59}$)
begin
  if pent$_5$ = p-error-cc_findings **then** a$_{59}$:= error-tfindings **else**
    a$_{59}$:= p-findings(pent$_5$)
end

reprort#(pent$_5$; **var** a$_{60}$)
begin
  if pent$_5$ = p-error-cc_findings **then** a$_{60}$:= error-letter **else**
    a$_{60}$:= p-reprort(pent$_5$)
end

set-findings_id#(pent$_5$, a$_{57}$; **var** pent5-$_1$)
begin
  if pent$_5$ = p-error-cc_findings $\vee$ a$_{57}$ = undef-findingsids $\vee$ a$_{57}$ = error-findingsids **then**
    pent5-$_1$:= p-error-cc_findings
  else
    pent5-$_1$:= p-mk-cc_findings(a$_{57}$,
      p-findingsdate(pent$_5$),
      p-findings(pent$_5$),
      p-reprort(pent$_5$))
end

set-findingsdate#(pent$_5$, a$_{58}$; **var** pent5-$_1$)
begin
  if pent$_5$ = p-error-cc_findings $\vee$ a$_{58}$ = undef-dates $\vee$ a$_{58}$ = error-dates **then**
    pent5-$_1$:= p-error-cc_findings
  else
    pent5-$_1$:= p-mk-cc_findings(p-findings_id(pent$_5$),
      a$_{58}$,
      p-findings(pent$_5$),
      p-reprort(pent$_5$))
end

```
set-findings#(pent₅, a₅₉; var pent5-₁)
begin
   if pent₅ = p-error-cc_findings ∨ a₅₉ = undef-tfindings ∨ a₅₉ = error-tfindings then
     pent5-₁ := p-error-cc_findings
   else
     pent5-₁ := p-mk-cc_findings(p-findings_id(pent₅),
                                 p-findingsdate(pent₅),
                                 a₅₉,
                                 p-reprort(pent₅))
end


set-reprort#(pent₅, a₆₀; var pent5-₁)
begin
   if pent₅ = p-error-cc_findings ∨ a₆₀ = error-letter then
     pent5-₁ := p-error-cc_findings
   else
     pent5-₁ := p-mk-cc_findings(p-findings_id(pent₅),
                                 p-findingsdate(pent₅),
                                 p-findings(pent₅),
                                 a₆₀)
end


key-cc_findings#(pent₅; var pkey₅)
begin
   if pent₅ = p-error-cc_findings then pkey₅ := p-mkkey-cc_findings(error-findings_id) else
     pkey₅ := p-mkkey-cc_findings(p-findings_id₂(pent₄))
end


mkkey-cc_findings#(a₅₇; var pkey₅)
begin
   pkey₅ := p-mkkey-cc_findings(a₅₇)
end


key-cc_findings≪#(pkey₅, pkey5-₁; var b)
begin
   if k-findings_id(pkey₅) ≪_findingsids k-findings_id(pkey5-₁) then b := tt else b := ff
end


rs-cc_findings#(pent₅)
begin
   if pent₅ = p-error-cc_findings then skip else
     var pent5-₁ = p-error-cc_findings in
     begin
        create-cc_findings#(p-findings_id(pent₅),
                            p-findingsdate(pent₅),
                            p-findings(pent₅),
                            p-reprort(pent₅);
                            pent5-₁);
        if pent5-₁ = p-error-cc_findings then abort
     end
end
```

rs-key-cc_findings#($pkey_5$)
begin skip end

A.2.5 Doctor

The Specifications

The Specification preDoctor An instantiation of the scheme $preEntity_i$.

```
preDoctor =
data specification
    using OrderedAttributes
    pre-doctor = p-mk-doctor (p-doctor_id : doctorids,
                              p-name2 : names,
                              p-address2 : adr,
                              p-rank : trank,
                              p-ward2 : wards,
                              p-entry : time,
                              p-leaving : time,
                              p-role : roles)
               | p-error-doctor
               ;
    p-keysort-doctor = p-mkkey-doctor (k-doctor_id : doctorids);
    variables
        pent2-1, pent2: pre-doctor;
        pkey2-1, pkey2: p-keysort-doctor;
end data specification
```

The Specification Doctor An instantiation of the scheme $Entity_i$.

```
Doctor =
enrich OrderedAttributes with
    sorts doctor, keysort-doctor;
    constants error-doctor : doctor;
    functions
        create-doctor   : doctorids × names × adr × trank
                          × wards × time × time × roles   →  doctor          ;
        doctor_id       : doctor                          →  doctorids       ;
        name2           : doctor                          →  names           ;
        address2        : doctor                          →  adr             ;
        rank            : doctor                          →  trank           ;
        ward2           : doctor                          →  wards           ;
        entry           : doctor                          →  time            ;
        leaving         : doctor                          →  time            ;
        role            : doctor                          →  roles           ;
        set-doctor_id   : doctor × doctorids              →  doctor          ;
        set-name2       : doctor × names                  →  doctor          ;
        set-address2    : doctor × adr                    →  doctor          ;
        set-rank        : doctor × trank                  →  doctor          ;
        set-ward2       : doctor × wards                  →  doctor          ;
        set-entry       : doctor × time                   →  doctor          ;
        set-leaving     : doctor × time                   →  doctor          ;
        set-role        : doctor × roles                  →  doctor          ;
        key-doctor      : doctor                          →  keysort-doctor  ;
        mkkey-doctor    : doctorids                       →  keysort-doctor  ;
    predicates . ≪key-doctor . : keysort-doctor × keysort-doctor;
    variables
```

ent_2: doctor;
key2-$_2$, key2-$_1$, key_2: keysort-doctor;
a_{29}, a_{21}: doctorids;
a_{30}, a_{22}: names;
a_{31}, a_{23}: adr;
a_{32}, a_{24}: trank;
a_{33}, a_{25}: wards;
a_{35}, a_{34}, a_{27}, a_{26}: time;
a_{36}, a_{28}: roles;

axioms

doctor **generated by** create-doctor, error-doctor;
keysort-doctor **freely generated by** mkkey-doctor;
a_{21} = undef-doctorids
$\vee$ a_{22} = undef-names
$\vee$ a_{23} = undef-adr
$\vee$ a_{24} = undef-trank
$\vee$ a_{26} = undef-time
$\vee$ a_{28} = undef-roles
$\vee$ a_{21} = error-doctorids
$\vee$ a_{22} = error-names
$\vee$ a_{23} = error-adr
$\vee$ a_{24} = error-trank
$\vee$ a_{25} = error-wards
$\vee$ a_{26} = error-time
$\vee$ a_{27} = error-time
$\vee$ a_{28} = error-roles
$\leftrightarrow$
create-doctor(a_{21}, a_{22}, a_{23}, a_{24}, a_{25}, a_{26}, a_{27}, a_{28}) = error-doctor,
create-doctor(a_{21}, a_{22}, a_{23}, a_{24}, a_{25}, a_{26}, a_{27}, a_{28}) $\neq$ error-doctor
$\rightarrow$ (create-doctor(a_{21}, a_{22}, a_{23}, a_{24}, a_{25}, a_{26}, a_{27}, a_{28})
= create-doctor(a_{29}, a_{30}, a_{31}, a_{32}, a_{33}, a_{34}, a_{35}, a_{36})
$\rightarrow$ $a_{21} = a_{29}$
$\wedge$ $a_{22} = a_{30}$
$\wedge$ $a_{23} = a_{31}$
$\wedge$ $a_{24} = a_{32}$
$\wedge$ $a_{25} = a_{33}$
$\wedge$ $a_{26} = a_{34}$
$\wedge$ $a_{27} = a_{35}$
$\wedge$ $a_{28} = a_{36}$)
$\wedge$ doctor_id(create-doctor(a_{21}, a_{22}, a_{23}, a_{24}, a_{25}, a_{26}, a_{27}, a_{28})) = a_{21}
$\wedge$ $name_2$(create-doctor(a_{21}, a_{22}, a_{23}, a_{24}, a_{25}, a_{26}, a_{27}, a_{28})) = a_{22}
$\wedge$ $address_2$(create-doctor(a_{21}, a_{22}, a_{23}, a_{24}, a_{25}, a_{26}, a_{27}, a_{28})) = a_{23}
$\wedge$ rank(create-doctor(a_{21}, a_{22}, a_{23}, a_{24}, a_{25}, a_{26}, a_{27}, a_{28})) = a_{24}
$\wedge$ $ward_2$(create-doctor(a_{21}, a_{22}, a_{23}, a_{24}, a_{25}, a_{26}, a_{27}, a_{28})) = a_{25}
$\wedge$ entry(create-doctor(a_{21}, a_{22}, a_{23}, a_{24}, a_{25}, a_{26}, a_{27}, a_{28})) = a_{26}
$\wedge$ leaving(create-doctor(a_{21}, a_{22}, a_{23}, a_{24}, a_{25}, a_{26}, a_{27}, a_{28})) = a_{27}
$\wedge$ role(create-doctor(a_{21}, a_{22}, a_{23}, a_{24}, a_{25}, a_{26}, a_{27}, a_{28})) = a_{28},
ent_2 $\neq$ error-doctor
$\rightarrow$ set-doctor_id(ent_2, a_{21})
= create-doctor(a_{21},
$name_2(ent_2)$,
$address_2(ent_2)$,
rank(ent_2),
$ward_2(ent_2)$,
entry(ent_2),
leaving(ent_2),

$$
\begin{aligned}
&\quad\quad\quad\quad \text{role}(\text{ent}_2)) \\
&\wedge\ \text{set-name}_2(\text{ent}_2,\ a_{22}) \\
&\quad = \text{create-doctor}(\text{doctor\_id}(\text{ent}_2), a_{22}, \text{address}_2(\text{ent}_2), \text{rank}(\text{ent}_2), \text{ward}_2(\text{ent}_2), \text{entry}(\text{ent}_2), \text{leaving}(\text{ent}_2), \text{role}(\text{ent}_2)) \\
&\wedge\ \text{set-address}_2(\text{ent}_2,\ a_{23}) \\
&\quad = \text{create-doctor}(\text{doctor\_id}(\text{ent}_2), \text{name}_2(\text{ent}_2), a_{23}, \text{rank}(\text{ent}_2), \text{ward}_2(\text{ent}_2), \text{entry}(\text{ent}_2), \text{leaving}(\text{ent}_2), \text{role}(\text{ent}_2)) \\
&\wedge\ \text{set-rank}(\text{ent}_2,\ a_{24}) \\
&\quad = \text{create-doctor}(\text{doctor\_id}(\text{ent}_2), \text{name}_2(\text{ent}_2), \text{address}_2(\text{ent}_2), a_{24}, \text{ward}_2(\text{ent}_2), \text{entry}(\text{ent}_2), \text{leaving}(\text{ent}_2), \text{role}(\text{ent}_2)) \\
&\wedge\ \text{set-ward}_2(\text{ent}_2,\ a_{25}) \\
&\quad = \text{create-doctor}(\text{doctor\_id}(\text{ent}_2), \text{name}_2(\text{ent}_2), \text{address}_2(\text{ent}_2), \text{rank}(\text{ent}_2), a_{25}, \text{entry}(\text{ent}_2), \text{leaving}(\text{ent}_2), \text{role}(\text{ent}_2)) \\
&\wedge\ \text{set-entry}(\text{ent}_2,\ a_{26}) \\
&\quad = \text{create-doctor}(\text{doctor\_id}(\text{ent}_2), \text{name}_2(\text{ent}_2), \text{address}_2(\text{ent}_2), \text{rank}(\text{ent}_2), \text{ward}_2(\text{ent}_2), a_{26}, \text{leaving}(\text{ent}_2), \text{role}(\text{ent}_2)) \\
&\wedge\ \text{set-leaving}(\text{ent}_2,\ a_{27}) \\
&\quad = \text{create-doctor}(\text{doctor\_id}(\text{ent}_2), \text{name}_2(\text{ent}_2), \text{address}_2(\text{ent}_2), \text{rank}(\text{ent}_2), \text{ward}_2(\text{ent}_2), \text{entry}(\text{ent}_2), a_{27}, \text{role}(\text{ent}_2)) \\
&\wedge\ \text{set-role}(\text{ent}_2,\ a_{28})
\end{aligned}
$$

$= \text{create-doctor}(\text{doctor\_id}(\text{ent}_2),$
$\text{name}_2(\text{ent}_2),$
$\text{address}_2(\text{ent}_2),$
$\text{rank}(\text{ent}_2),$
$\text{ward}_2(\text{ent}_2),$
$\text{entry}(\text{ent}_2),$
$\text{leaving}(\text{ent}_2),$
$\text{a}_{28}),$

doctor_id(error-doctor) = error-doctorids,
name_2(error-doctor) = error-names,
address_2(error-doctor) = error-adr,
rank(error-doctor) = error-trank,
ward_2(error-doctor) = error-wards,
entry(error-doctor) = error-time,
leaving(error-doctor) = error-time,
role(error-doctor) = error-roles,
set-doctor_id(error-doctor, a_{21}) = error-doctor,
set-name_2(error-doctor, a_{22}) = error-doctor,
set-address_2(error-doctor, a_{23}) = error-doctor,
set-rank(error-doctor, a_{24}) = error-doctor,
set-ward_2(error-doctor, a_{25}) = error-doctor,
set-entry(error-doctor, a_{26}) = error-doctor,
set-leaving(error-doctor, a_{27}) = error-doctor,
set-role(error-doctor, a_{28}) = error-doctor,
key-doctor(ent_2) = mkkey-doctor(doctor_id(ent_2)),
$\text{mkkey-doctor}(\text{a}_{21}) \ll_{key-doctor} \text{mkkey-doctor}(\text{a}_{29}) \leftrightarrow \text{a}_{21} \ll_{doctorids} \text{a}_{29},$
$\neg\ \text{key}_2 \ll_{key-doctor} \text{key}_2,$
$\text{key}_2 \ll_{key-doctor} \text{key2-}_1 \wedge \text{key2-}_1 \ll_{key-doctor} \text{key2-}_2 \rightarrow \text{key}_2 \ll_{key-doctor} \text{key2-}_2,$
$\text{key}_2 \ll_{key-doctor} \text{key2-}_1 \vee \text{key}_2 = \text{key2-}_1 \vee \text{key2-}_1 \ll_{key-doctor} \text{key}_2$

end enrich

The Implementation

Entity-preEntity-Doctor =
module
export Doctor
refinement
representation of sorts

pre-doctor	implements	doctor;
p-keysort-doctor	implements	keysort-doctor;

representation of operations

error-doctor#	implements	error-doctor;
create-doctor#	implements	create-doctor;
doctor_id#	implements	doctor_id;
name2#	implements	name_2;
address2#	implements	address_2;
rank#	implements	rank;
ward2#	implements	ward_2;
entry#	implements	entry;
leaving#	implements	leaving;
role#	implements	role;
set-doctor_id#	implements	set-doctor_id;
set-name2#	implements	set-name_2;
set-address2#	implements	set-address_2;
set-rank#	implements	set-rank;
set-ward2#	implements	set-ward_2;

set-entry#	implements	set-entry;
set-leaving#	implements	set-leaving;
set-role#	implements	set-role;
key-doctor#	implements	key-doctor;
mkkey-doctor#	implements	mkkey-doctor;
key-doctor≪#	implements	$\ll_{key-doctor}$;

import preDoctor

procedures

error-doctor#	()	: pre-doctor;
create-doctor#	(doctorids, names, adr, trank, wards, time, time, roles)	: pre-doctor;
doctor_id#	(pre-doctor)	: doctorids;
name2#	(pre-doctor)	: names;
address2#	(pre-doctor)	: adr;
rank#	(pre-doctor)	: trank;
ward2#	(pre-doctor)	: wards;
entry#	(pre-doctor)	: time;
leaving#	(pre-doctor)	: time;
role#	(pre-doctor)	: roles;
set-doctor_id#	(pre-doctor, doctorids)	: pre-doctor;
set-name2#	(pre-doctor, names)	: pre-doctor;
set-address2#	(pre-doctor, adr)	: pre-doctor;
set-rank#	(pre-doctor, trank)	: pre-doctor;
set-ward2#	(pre-doctor, wards)	: pre-doctor;
set-entry#	(pre-doctor, time)	: pre-doctor;
set-leaving#	(pre-doctor, time)	: pre-doctor;
set-role#	(pre-doctor, roles)	: pre-doctor;
key-doctor#	(pre-doctor)	: p-keysort-doctor;
mkkey-doctor#	(doctorids)	: p-keysort-doctor;
key-doctor≪#	(p-keysort-doctor, p-keysort-doctor)	: bool;

variables

b: bool;
a_{21}: doctorids;
a_{22}: names;
a_{23}: adr;
a_{24}: trank;
a_{25}: wards;
a_{27}, a_{26}: time;
a_{28}: roles;

implementation

error-doctor#(**var** $pent_2$)
begin
  $pent_2$:= p-error-doctor
end

create-doctor#(a_{21}, a_{22}, a_{23}, a_{24}, a_{25}, a_{26}, a_{27}, a_{28}; **var** $pent_2$)
begin
  if a_{21} = undef-doctorids

```
      ∨ a22 = undef-names
      ∨ a23 = undef-adr
      ∨ a24 = undef-trank
      ∨ a26 = undef-time
      ∨ a28 = undef-roles
      ∨ a21 = error-doctorids
      ∨ a22 = error-names
      ∨ a23 = error-adr
      ∨ a24 = error-trank
      ∨ a25 = error-wards
      ∨ a26 = error-time
      ∨ a27 = error-time
      ∨ a28 = error-roles then
     pent2 := p-error-doctor
  else
     pent2 := p-mk-doctor(a21, a22, a23, a24, a25, a26, a27, a28)
end


doctor_id#(pent2; var a21)
begin
  if pent2 = p-error-doctor then a21 := error-doctorids else a21 := p-doctor_id(pent2)
end


name2#(pent2; var a22)
begin
  if pent2 = p-error-doctor then a22 := error-names else a22 := p-name2(pent2)
end


address2#(pent2; var a23)
begin
  if pent2 = p-error-doctor then a23 := error-adr else a23 := p-address2(pent2)
end


rank#(pent2; var a24)
begin
  if pent2 = p-error-doctor then a24 := error-trank else a24 := p-rank(pent2)
end


ward2#(pent2; var a25)
begin
  if pent2 = p-error-doctor then a25 := error-wards else a25 := p-ward2(pent2)
end


entry#(pent2; var a26)
begin
  if pent2 = p-error-doctor then a26 := error-time else a26 := p-entry(pent2)
end
```

leaving#($pent_2$; **var** a_{27})
begin
  if $pent_2$ = p-error-doctor **then** a_{27} := error-time **else** a_{27} := p-leaving($pent_2$)
end

role#($pent_2$; **var** a_{28})
begin
  if $pent_2$ = p-error-doctor **then** a_{28} := error-roles **else** a_{28} := p-role($pent_2$)
end

set-doctor_id#($pent_2$, a_{21}; **var** pent2-$_1$)
begin
  if $pent_2$ = p-error-doctor $\vee$ a_{21} = undef-doctorids $\vee$ a_{21} = error-doctorids **then**
    pent2-$_1$:= p-error-doctor
  else
    pent2-$_1$:= p-mk-doctor(a_{21},
      p-$name_2$($pent_2$),
      p-$address_2$($pent_2$),
      p-rank($pent_2$),
      p-$ward_2$($pent_2$),
      p-entry($pent_2$),
      p-leaving($pent_2$),
      p-role($pent_2$))
end

set-name2#($pent_2$, a_{22}; **var** pent2-$_1$)
begin
  if $pent_2$ = p-error-doctor $\vee$ a_{22} = undef-names $\vee$ a_{22} = error-names **then**
    pent2-$_1$:= p-error-doctor
  else
    pent2-$_1$:= p-mk-doctor(p-doctor_id($pent_2$),
      a_{22},
      p-$address_2$($pent_2$),
      p-rank($pent_2$),
      p-$ward_2$($pent_2$),
      p-entry($pent_2$),
      p-leaving($pent_2$),
      p-role($pent_2$))
end

set-address2#($pent_2$, a_{23}; **var** pent2-$_1$)
begin
  if $pent_2$ = p-error-doctor $\vee$ a_{23} = undef-adr $\vee$ a_{23} = error-adr **then**
    pent2-$_1$:= p-error-doctor
  else
    pent2-$_1$:= p-mk-doctor(p-doctor_id($pent_2$),
      p-$name_2$($pent_2$),
      a_{23},
      p-rank($pent_2$),
      p-$ward_2$($pent_2$),

```
                                    p-entry(pent_2),
                                    p-leaving(pent_2),
                                    p-role(pent_2))
end


set-rank#(pent_2, a_24; var pent2-_1)
begin
  if pent_2 = p-error-doctor ∨ a_24 = undef-trank ∨ a_24 = error-trank then
    pent2-_1 := p-error-doctor
  else
    pent2-_1 := p-mk-doctor(p-doctor_id(pent_2),
                                    p-name_2(pent_2),
                                    p-address_2(pent_2),
                                    a_24,
                                    p-ward_2(pent_2),
                                    p-entry(pent_2),
                                    p-leaving(pent_2),
                                    p-role(pent_2))
end


set-ward2#(pent_2, a_25; var pent2-_1)
begin
  if pent_2 = p-error-doctor ∨ a_25 = error-wards then
    pent2-_1 := p-error-doctor
  else
    pent2-_1 := p-mk-doctor(p-doctor_id(pent_2),
                                    p-name_2(pent_2),
                                    p-address_2(pent_2),
                                    p-rank(pent_2),
                                    a_25,
                                    p-entry(pent_2),
                                    p-leaving(pent_2),
                                    p-role(pent_2))
end


set-entry#(pent_2, a_26; var pent2-_1)
begin
  if pent_2 = p-error-doctor ∨ a_26 = undef-time ∨ a_26 = error-time then
    pent2-_1 := p-error-doctor
  else
    pent2-_1 := p-mk-doctor(p-doctor_id(pent_2),
                                    p-name_2(pent_2),
                                    p-address_2(pent_2),
                                    p-rank(pent_2),
                                    p-ward_2(pent_2),
                                    a_26,
                                    p-leaving(pent_2),
                                    p-role(pent_2))
end
```

```
set-leaving#(pent2, a27; var pent2-1)
begin
  if pent2 = p-error-doctor ∨ a27 = error-time then
    pent2-1 := p-error-doctor
  else
    pent2-1 := p-mk-doctor(p-doctor_id(pent2),
                           p-name2(pent2),
                           p-address2(pent2),
                           p-rank(pent2),
                           p-ward2(pent2),
                           p-entry(pent2),
                           a27,
                           p-role(pent2))
end
```

```
set-role#(pent2, a28; var pent2-1)
begin
  if pent2 = p-error-doctor ∨ a28 = error-roles then
    pent2-1 := p-error-doctor
  else
    pent2-1 := p-mk-doctor(p-doctor_id(pent2),
                           p-name2(pent2),
                           p-address2(pent2),
                           p-rank(pent2),
                           p-ward2(pent2),
                           p-entry(pent2),
                           p-leaving(pent2),
                           a28)
end
```

```
key-doctor#(pent2; var pkey2)
begin
  if pent2 = p-error-doctor then pkey2 := p-mkkey-doctor(p-error-doctor_id) else
    pkey2 := p-mkkey-doctor(p-doctor_id(pent2))
end
```

```
mkkey-doctor#(a21; var pkey2)
begin
  pkey2 := p-mkkey-doctor(a21)
end
```

```
key-doctor≪#(pkey2, pkey2-1; var b)
begin
  if k-doctor_id(pkey2) ≪doctorids k-doctor_id(pkey2-1) then b := tt else b := ff
end
```

```
rs-doctor#(pent2)
begin
  if pent2 = p-error-doctor then skip else
    var pent2-1 = p-error-doctor in
    begin
      create-doctor#(p-doctor_id(pent2),
                     p-name2(pent2),
                     p-address2(pent2),
                     p-rank(pent2),
                     p-ward2(pent2),
                     p-entry(pent2),
                     p-leaving(pent2),
                     p-role(pent2);
                     pent2-1);
      if pent2-1 = p-error-doctor then abort
    end
end
```

```
rs-key-doctor#(pkey2)
begin skip end
```

A.3 The Entity Sets

A.3.1 The Specification set-Patient

An instantiation of the scheme set-Entity$_i$.

set-Patient =

actualize coded-set **with** Patient **by morphism**

coded-set → set-patient, element → patient, key → keysort-patient, error-elem → error-patient, encode → key-patient, $\ll_{key}$ → $\ll_{key-patient}$, an-elem → ent$_1$, a-key → key$_1$, a-key$_1$ → key1-$_1$, a-key$_2$ → key1-$_2$, empty-c-set → emptyset-patient, error-c-set → errorset-patient, $+_{cs}$ → $+_{patient}$, $-_{cs}$ → $-_{patient}$, sel → sel-patient, min-cs → min-patient, rest-cs → rest-patient, in-cs → in-patient, realsub-cs → rsub-patient, el → ent1-$_0$, el$_1$ → ent1-$_1$, el$_2$ → ent1-$_2$, cs → sent$_1$, cs$_1$ → sent1-$_1$, cs$_2$ → sent1-$_2$

end actualize

A.3.2 The Specification set-CC_OR

An instantiation of the scheme set-Entity$_i$.

set-CC_OR =

actualize coded-set **with** CC_OR **by morphism**

coded-set → set-cc_or, element → cc_or, key → keysort-cc_or, error-elem → error-cc_or, encode → key-cc_or, $\ll_{key}$ → $\ll_{key-cc\_or}$, an-elem → ent$_3$, a-key → key$_3$, a-key$_1$ → key3-$_1$, a-key$_2$ → key3-$_2$, empty-c-set → emptyset-cc_or, error-c-set → errorset-cc_or, $+_{cs}$ → $+_{cc\_or}$, $-_{cs}$ → $-_{cc\_or}$, sel → sel-cc_or, min-cs → min-cc_or, rest-cs → rest-cc_or, in-cs → in-cc_or, realsub-cs → rsub-cc_or, el → ent3-$_0$, el$_1$ → ent3-$_1$, el$_2$ → ent3-$_2$, cs → sent$_3$, cs$_1$ → sent3-$_1$, cs$_2$ → sent3-$_2$

end actualize

A.3.3 The Specification set-CC_Data

An instantiation of the scheme set-Entity$_i$.

set-CC_Data =

actualize coded-set **with** CC_Data **by morphism**

coded-set $\rightarrow$ set-cc_data, element $\rightarrow$ cc_data, key $\rightarrow$ keysort-cc_data, error-elem $\rightarrow$ error-cc_data, encode $\rightarrow$ key-cc_data, $\ll_{key} \rightarrow \ll_{key-cc\_data}$, an-elem $\rightarrow$ ent$_4$, a-key $\rightarrow$ key$_4$, a-key$_1$ $\rightarrow$ key4-$_1$, a-key$_2$ $\rightarrow$ key4-$_2$, empty-c-set $\rightarrow$ emptyset-cc_data, error-c-set $\rightarrow$ errorset-cc_data, $+_{cs} \rightarrow +_{cc\_data}$, $-_{cs} \rightarrow -_{cc\_data}$, sel $\rightarrow$ sel-cc_data, min-cs $\rightarrow$ min-cc_data, rest-cs $\rightarrow$ rest-cc_data, in-cs $\rightarrow$ in-cc_data, realsub-cs $\rightarrow$ rsub-cc_data, el $\rightarrow$ ent4-$_0$, el$_1$ $\rightarrow$ ent4-$_1$, el$_2$ $\rightarrow$ ent4-$_2$, cs $\rightarrow$ sent$_4$, cs$_1$ $\rightarrow$ sent4-$_1$, cs$_2$ $\rightarrow$ sent4-$_2$

end actualize

A.3.4 The Specification set-CC_Findings

An instantiation of the scheme set-Entity$_i$.

set-CC_Findings =

actualize coded-set **with** CC_Findings **by morphism**

coded-set $\rightarrow$ set-cc_findings, element $\rightarrow$ cc_findings, key $\rightarrow$ keysort-cc_findings, error-elem $\rightarrow$ error-cc_findings, encode $\rightarrow$ key-cc_findings, $\ll_{key} \rightarrow \ll_{key-cc\_findings}$, an-elem $\rightarrow$ ent$_5$, a-key $\rightarrow$ key$_5$, a-key$_1$ $\rightarrow$ key5-$_1$, a-key$_2$ $\rightarrow$ key5-$_2$, empty-c-set $\rightarrow$ emptyset-cc_findings, error-c-set $\rightarrow$ errorset-cc_findings, $+_{cs} \rightarrow +_{cc\_findings}$, $-_{cs} \rightarrow -_{cc\_findings}$, sel $\rightarrow$ sel-cc_findings, min-cs $\rightarrow$ min-cc_findings, rest-cs $\rightarrow$ rest-cc_findings, in-cs $\rightarrow$ in-cc_findings, realsub-cs $\rightarrow$ rsub-cc_findings, el $\rightarrow$ ent5-$_0$, el$_1$ $\rightarrow$ ent5-$_1$, el$_2$ $\rightarrow$ ent5-$_2$, cs $\rightarrow$ sent$_5$, cs$_1$ $\rightarrow$ sent5-$_1$, cs$_2$ $\rightarrow$ sent5-$_2$

end actualize

A.3.5 The Specification set-Doctor

An instantiation of the scheme set-Entity$_i$.

set-Doctor =

actualize coded-set **with** Doctor **by morphism**

coded-set $\rightarrow$ set-doctor, element $\rightarrow$ doctor, key $\rightarrow$ keysort-doctor, error-elem $\rightarrow$ error-doctor, encode $\rightarrow$ key-doctor, $\ll_{key} \rightarrow \ll_{key-doctor}$, an-elem $\rightarrow$ ent$_2$, a-key $\rightarrow$ key$_2$, a-key$_1$ $\rightarrow$ key2-$_1$, a-key$_2$ $\rightarrow$ key2-$_2$, empty-c-set $\rightarrow$ emptyset-doctor, error-c-set $\rightarrow$ errorset-doctor, $+_{cs} \rightarrow +_{doctor}$, $-_{cs} \rightarrow -_{doctor}$, sel $\rightarrow$ sel-doctor, min-cs $\rightarrow$ min-doctor, rest-cs $\rightarrow$ rest-doctor, in-cs $\rightarrow$ in-doctor, realsub-cs $\rightarrow$ rsub-doctor, el $\rightarrow$ ent2-$_0$, el$_1$ $\rightarrow$ ent2-$_1$, el$_2$ $\rightarrow$ ent2-$_2$, cs $\rightarrow$ sent$_2$, cs$_1$ $\rightarrow$ sent2-$_1$, cs$_2$ $\rightarrow$ sent2-$_2$

end actualize

A.4 The Relations

A.4.1 The Specification part_of

At first the union between Patient and CC_OR an instantiation of the scheme Entity$_{r_{j1}}$_u_Entity$_{r_{j2}}$.

Patient_u_CC_OR = Patient + CC_OR

At second the instantiation of scheme R$_j$.

part_of =

actualize pair-orderset **with** Patient_u_CC_OR **by morphism**

pair-oset → reltype-part_of, elem1 → keysort-patient, elem2 → keysort-cc_or, $\ll_{elem1}$ → $\ll_{key-patient}$, $\ll_{elem2}$ → $\ll_{key-cc\_or}$, e_0 → k1-$_0$, e_1 → k1-$_1$, e1-$_1$ → k1-1-$_1$, e1-$_2$ → k1-1-$_2$, e_2 → k1-$_2$, e2-$_1$ → k1-2-$_1$, e2-$_2$ → k1-2-$_2$, e_3 → k1-$_3$, pair → pair-part_of, $\ll_{pair}$ → $\ll_{part\_of}$, mkpair → mk-part_of, .fst → fst-part_of, .snd → snd-part_of, par_1 → k1-$_1$, par_2 → k1-$_2$, p → $pair_1$, p_1 → pair1-$_1$, p_2 → pair1-$_2$, empty-p-set → emptyrel-part_of, $+_{ps}$ → $+_{part\_of}$, $-_{ps}$ → $-_{part\_of}$, min-ps → min-part_of, rest-ps → rest-part_of, in-ps → in-part_of, realsub-ps → rsub-part_of, ap → $pair_1$, ap_1 → pair1-$_1$, ap_2 → pair1-$_2$, ps → rel_1, ps_1 → rel1-$_1$, ps_2 → rel1-$_2$
end actualize

A.4.2 The Specification orders

At first the union between Doctor and CC_OR an instantiation of the scheme $Entity_{r_{j1}}$_u_$Entity_{r_{j2}}$.

Doctor_u_CC_OR = Doctor + CC_OR

At second the instantiation of scheme R_j.

orders =
actualize pair-orderset **with** Doctor_u_CC_OR **by morphism**
pair-oset → reltype-orders, elem1 → keysort-doctor, elem2 → keysort-cc_or, $\ll_{elem1}$ → $\ll_{key-doctor}$, $\ll_{elem2}$ → $\ll_{key-cc\_or}$, e_0 → k2-$_0$, e_1 → k2-$_1$, e1-$_1$ → k2-1-$_1$, e1-$_2$ → k2-1-$_2$, e_2 → k2-$_2$, e2-$_1$ → k2-2-$_1$, e2-$_2$ → k2-2-$_2$, e_3 → k2-$_3$, pair → pair-orders, $\ll_{pair}$ → $\ll_{orders}$, mkpair → mk-orders, .fst → fst-orders, .snd → snd-orders, par_1 → k2-$_1$, par_2 → k2-$_2$, p → $pair_2$, p_1 → pair2-$_1$, p_2 → pair2-$_2$, empty-p-set → emptyrel-orders, $+_{ps}$ → $+_{orders}$, $-_{ps}$ → $-_{orders}$, min-ps → min-orders, rest-ps → rest-orders, in-ps → in-orders, realsub-ps → rsub-orders, ap → $pair_2$, ap_1 → pair2-$_1$, ap_2 → pair2-$_2$, ps → rel_2, ps_1 → rel2-$_1$, ps_2 → rel2-$_2$
end actualize

A.4.3 The Specification examination

At first the union between CC_OR and CC_Data an instantiation of the scheme $Entity_{r_{j1}}$_u_$Entity_{r_{j2}}$.

CC_OR_u_CC_Data = CC_OR + CC_Data

At second the instantiation of scheme R_j.

examination =
actualize pair-orderset **with** CC_OR_u_CC_Data **by morphism**
pair-oset → reltype-examination, elem1 → keysort-cc_or, elem2 → keysort-cc_data, $\ll_{elem1}$ → $\ll_{key-cc\_or}$, $\ll_{elem2}$ → $\ll_{key-cc\_data}$, e_0 → k5-$_0$, e_1 → k5-$_1$, e1-$_1$ → k5-1-$_1$, e1-$_2$ → k5-1-$_2$, e_2 → k5-$_2$, e2-$_1$ → k5-2-$_1$, e2-$_2$ → k5-2-$_2$, e_3 → k5-$_3$, pair → pair-examination, $\ll_{pair}$ → $\ll_{examination}$, mkpair → mk-examination, .fst → fst-examination, .snd → snd-examination, par_1 → k5-$_1$, par_2 → k5-$_2$, p → $pair_5$, p_1 → pair5-$_1$, p_2 → pair5-$_2$, empty-p-set → emptyrel-examination, $+_{ps}$ → $+_{examination}$, $-_{ps}$ → $-_{examination}$, min-ps → min-examination, rest-ps → rest-examination, in-ps → in-examination, realsub-ps → rsub-examination, ap → $pair_5$, ap_1 → pair5-$_1$, ap_2 → pair5-$_2$, ps → rel_5, ps_1 → rel5-$_1$, ps_2 → rel5-$_2$
end actualize

A.4.4 The Specification determine

At first the union between Doctor and CC_Data an instantiation of the scheme Entity_{rj1}_u_Entity_{rj2}.

Doctor_u_CC_Data = Doctor + CC_Data

At second the instantiation of scheme R_j.

determine =

actualize pair-orderset **with** Doctor_u_CC_Data **by morphism**
pair-oset → reltype-determine, elem1 → keysort-doctor, elem2 → keysort-cc_data, $\ll_{elem1} \rightarrow \ll_{key-doctor}$, $\ll_{elem2} \rightarrow \ll_{key-cc\_data}$, $e_0 \rightarrow$ k3-$_0$, $e_1 \rightarrow$ k3-$_1$, e1-$_1$ → k3-1-$_1$, e1-$_2$ → k3-1-$_2$, e_2 → k3-$_2$, e2-$_1$ → k3-2-$_1$, e2-$_2$ → k3-2-$_2$, e_3 → k3-$_3$, pair → pair-determine, $\ll_{pair} \rightarrow \ll_{determine}$, mkpair → mk-determine, .fst → fst-determine, .snd → snd-determine, par_1 → k3-$_1$, par_2 → k3-$_2$, p → pair_3, p_1 → pair3-$_1$, p_2 → pair3-$_2$, empty-p-set → emptyrel-determine, $+_{ps} \rightarrow +_{determine}$, $-_{ps} \rightarrow -_{determine}$, min-ps → min-determine, rest-ps → rest-determine, in-ps → in-determine, realsub-ps → rsub-determine, ap → pair_3, ap_1 → pair3-$_1$, ap_2 → pair3-$_2$, ps → rel_3, ps_1 → rel3-$_1$, ps_2 → rel3-$_2$
end actualize

A.4.5 The Specification make

At first the union between Doctor and CC_Findings an instantiation of the scheme Entity_{rj1}_u_Entity_{rj2}.

Doctor_u_CC_Findings = Doctor + CC_Findings

At second the instantiation of scheme R_j.

make =

actualize pair-orderset **with** Doctor_u_CC_Findings **by morphism**
pair-oset → reltype-make, elem1 → keysort-doctor, elem2 → keysort-cc_findings, $\ll_{elem1} \rightarrow \ll_{key-doctor}$, $\ll_{elem2} \rightarrow \ll_{key-cc\_findings}$, e_0 → k4-$_0$, e_1 → k4-$_1$, e1-$_1$ → k4-1-$_1$, e1-$_2$ → k4-1-$_2$, e_2 → k4-$_2$, e2-$_1$ → k4-2-$_1$, e2-$_2$ → k4-2-$_2$, e_3 → k4-$_3$, pair → pair-make, $\ll_{pair} \rightarrow \ll_{make}$, mkpair → mk-make, .fst → fst-make, .snd → snd-make, par_1 → k4-$_1$, par_2 → k4-$_2$, p → pair_4, p_1 → pair4-$_1$, p_2 → pair4-$_2$, empty-p-set → emptyrel-make, $+_{ps} \rightarrow +_{make}$, $-_{ps} \rightarrow -_{make}$, min-ps → min-make, rest-ps → rest-make, in-ps → in-make, realsub-ps → rsub-make, ap → pair_4, ap_1 → pair4-$_1$, ap_2 → pair4-$_2$, ps → rel_4, ps_1 → rel4-$_1$, ps_2 → rel4-$_2$
end actualize

A.4.6 The Specification finding

At first the union between CC_Data and CC_Findings an instantiation of the scheme Entity_{rj1}_u_Entity_{rj2}.

CC_Data_u_CC_Findings = CC_Data + CC_Findings

At second the instantiation of scheme R_j.

finding =

actualize pair-orderset **with** CC_Data_u_CC_Findings **by morphism**

pair-oset → reltype-finding, elem1 → keysort-cc_data, elem2 → keysort-cc_findings, $\ll_{elem1} \to \ll_{key-cc\_data}$, $\ll_{elem2} \to \ll_{key-cc\_findings}$, $e_0 \to$ k6-$_0$, $e_1 \to$ k6-$_1$, e1-$_1 \to$ k6-1-$_1$, e1-$_2 \to$ k6-1-$_2$, $e_2 \to$ k6-$_2$, e2-$_1 \to$ k6-2-$_1$, e2-$_2 \to$ k6-2-$_2$, $e_3 \to$ k6-$_3$, pair → pair-finding, $\ll_{pair} \to \ll_{finding}$, mkpair → mk-finding, .fst → fst-finding, .snd → snd-finding, $par_1 \to$ k6-$_1$, $par_2 \to$ k6-$_2$, p → $pair_6$, $p_1 \to$ pair6-$_1$, $p_2 \to$ pair6-$_2$, empty-p-set → emptyrel-finding, $+_{ps} \to +_{finding}$, $-_{ps} \to -_{finding}$, min-ps → min-finding, rest-ps → rest-finding, in-ps → in-finding, realsub-ps → rsub-finding, ap → $pair_6$, $ap_1 \to$ pair6-$_1$, $ap_2 \to$ pair6-$_2$, ps → rel_6, $ps_1 \to$ rel6-$_1$, $ps_2 \to$ rel6-$_2$

end actualize

A.5 The Database

A.5.1 The Specifications

The Step by Step Combination

1. The union of Entity sets:

   ENTITIESETS_Cardiac-Catheterisation = set-Patient
   + set-Doctor
   + set-CC_OR
   + set-CC_Data
   + set-CC_Findings

2. The union of Relations:

   RELATIONS_Cardiac-Catheterisation = part_of
   + orders
   + determine
   + make
   + examination
   + finding

3. the union of ENTITIESETS and RELATIONS:

   ESETS+REL_Cardiac-Catheterisation = ENTITIESETS_Cardiac-Catheterisation
   + RELATIONS_Cardiac-Catheterisation

At last we can present the database specification.

The Specification pre-DB_Cardiac-Catheterisation

An instantiation of the scheme pre-DB.

pre-DB_Cardiac-Catheterisation =
data specification
  using ESETS+REL_Cardiac-Catheterisation
  pre-db = p-mk-db (p-ent-patient : set-patient,
    p-ent-doctor : set-doctor,
    p-ent-cc_or : set-cc_or,
    p-ent-cc_data : set-cc_data,

```
                    p-ent-cc_findings : set-cc_findings,
                    p-part_of : reltype-part_of,
                    p-orders : reltype-orders,
                    p-determine : reltype-determine,
                    p-make : reltype-make,
                    p-examination : reltype-examination,
                    p-finding : reltype-finding)
               | p-error-db
               ;
  variables vpdb: pre-db;
end data specification
```

The Specification ER_DB_Cardiac-Catheterisation

An instantiation of the scheme ER_DB.

```
ER_DB_Cardiac-Catheterisation =
enrich ESETS+REL_Cardiac-Catheterisation with
  sorts db;
  constants empty-db : db; error-db : db;
  functions
     mk-db             :  set-patient
                          × set-doctor
                          × set-cc_or
                          × set-cc_data
                          × set-cc_findings
                          × reltype-part_of
                          × reltype-orders
                          × reltype-determine
                          × reltype-make
                          × reltype-examination
                          × reltype-finding                →  db               ;
     ent-patient       :  db                               →  set-patient      ;
     ent-doctor        :  db                               →  set-doctor       ;
     ent-cc_or         :  db                               →  set-cc_or        ;
     ent-cc_data       :  db                               →  set-cc_data      ;
     ent-cc_findings   :  db                               →  set-cc_findings  ;
     put-patient       :  patient × db                     →  db               ;
     put-doctor        :  doctor × db                      →  db               ;
     put-cc_or         :  cc_or × db                       →  db               ;
     put-cc_data       :  cc_data × db                     →  db               ;
     put-cc_findings   :  cc_findings × db                 →  db               ;
     del-patient       :  patient × db                     →  db               ;
     del-doctor        :  doctor × db                      →  db               ;
     del-cc_or         :  cc_or × db                       →  db               ;
     del-cc_data       :  cc_data × db                     →  db               ;
     del-cc_findings   :  cc_findings × db                 →  db               ;
     get-patient       :  keysort-patient × db             →  patient          ;
     get-doctor        :  keysort-doctor × db              →  doctor           ;
     get-cc_or         :  keysort-cc_or × db               →  cc_or            ;
     get-cc_data       :  keysort-cc_data × db             →  cc_data          ;
     get-cc_findings   :  keysort-cc_findings × db         →  cc_findings      ;
     update-patient    :  keysort-patient × patient × db   →  db               ;
     update-doctor     :  keysort-doctor × doctor × db     →  db               ;
     update-cc_or      :  keysort-cc_or × cc_or × db       →  db               ;
     update-cc_data    :  keysort-cc_data × cc_data × db   →  db               ;
```

update-cc_findings : keysort-cc_findings $\times$ cc_findings $\times$ db $\rightarrow$ db ;
est-part_of : db $\times$ patient $\times$ cc_or $\rightarrow$ db ;
est-orders : db $\times$ doctor $\times$ cc_or $\rightarrow$ db ;
est-determine : db $\times$ doctor $\times$ cc_data $\rightarrow$ db ;
est-make : db $\times$ doctor $\times$ cc_findings $\rightarrow$ db ;
est-examination : db $\times$ cc_or $\times$ cc_data $\rightarrow$ db ;
est-finding : db $\times$ cc_data $\times$ cc_findings $\rightarrow$ db ;
rel-part_of : db $\times$ patient $\times$ cc_or $\rightarrow$ db ;
rel-orders : db $\times$ doctor $\times$ cc_or $\rightarrow$ db ;
rel-determine : db $\times$ doctor $\times$ cc_data $\rightarrow$ db ;
rel-make : db $\times$ doctor $\times$ cc_findings $\rightarrow$ db ;
rel-examination : db $\times$ cc_or $\times$ cc_data $\rightarrow$ db ;
rel-finding : db $\times$ cc_data $\times$ cc_findings $\rightarrow$ db ;

predicates

part_of : db $\times$ patient $\times$ cc_or;
orders : db $\times$ doctor $\times$ cc_or;
determine : db $\times$ doctor $\times$ cc_data;
make : db $\times$ doctor $\times$ cc_findings;
examination : db $\times$ cc_or $\times$ cc_data;
finding : db $\times$ cc_data $\times$ cc_findings;

variables vdb: db;

axioms

db **generated by** mk-db, error-db;

mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$, rel$_2$, rel$_3$, rel$_4$, rel$_5$, rel$_6$) $\neq$ error-db
$\leftrightarrow$
($\forall$ ent1-$_1$, ent1-$_2$. ent1-$_1$ in-patient sent$_1$ $\wedge$ ent1-$_2$ in-patient sent$_1$ $\wedge$ ent1-$_1$ $\neq$ ent1-$_2$
$\rightarrow$ key-patient(ent1-$_1$) $\neq$ key-patient(ent1-$_2$))
$\wedge$ ($\forall$ ent2-$_1$, ent2-$_2$. ent2-$_1$ in-doctor sent$_2$ $\wedge$ ent2-$_2$ in-doctor sent$_2$ $\wedge$ ent2-$_1$ $\neq$ ent2-$_2$
$\rightarrow$ key-doctor(ent2-$_1$) $\neq$ key-doctor(ent2-$_2$))
$\wedge$ ($\forall$ ent3-$_1$, ent3-$_2$. ent3-$_1$ in-cc_or sent$_3$ $\wedge$ ent3-$_2$ in-cc_or sent$_3$ $\wedge$ ent3-$_1$ $\neq$ ent3-$_2$
$\rightarrow$ key-cc_or(ent3-$_1$) $\neq$ key-cc_or(ent3-$_2$))
$\wedge$ ($\forall$ ent4-$_1$, ent4-$_2$. ent4-$_1$ in-cc_data sent$_4$ $\wedge$ ent4-$_2$ in-cc_data sent$_4$ $\wedge$ ent4-$_1$ $\neq$ ent4-$_2$
$\rightarrow$ key-cc_data(ent4-$_1$) $\neq$ key-cc_data(ent4-$_2$))
$\wedge$ ($\forall$ ent5-$_1$, ent5-$_2$. ent5-$_1$ in-cc_findings sent$_5$ $\wedge$ ent5-$_2$ in-cc_findings sent$_5$ $\wedge$ ent5-$_1$ $\neq$ ent5-$_2$
$\rightarrow$ key-cc_findings(ent5-$_1$) $\neq$ key-cc_findings(ent5-$_2$))
$\wedge$ ($\forall$ k1-$_1$, k1-$_2$. mk-part_of(k1-$_1$, k1-$_2$) in-part_of rel$_1$
$\rightarrow$ ($\exists$ ent1-$_1$,ent3-$_2$.ent1-$_1$ in-patient sent$_1$ $\wedge$ ent3-$_2$ in-cc_or sent$_3$
$\wedge$ key-patient(ent1-$_1$) = k1-$_1$ $\wedge$ key-cc_or(ent3-$_2$) = k1-$_2$))
$\wedge$ ($\forall$ k2-$_1$, k2-$_2$. mk-orders(k2-$_1$, k2-$_2$) in-orders rel$_2$
$\rightarrow$ ($\exists$ ent2-$_1$,ent3-$_2$.ent2-$_1$ in-doctor sent$_2$ $\wedge$ ent3-$_2$ in-cc_or sent$_3$
$\wedge$ key-doctor(ent2-$_1$) = k2-$_1$ $\wedge$ key-cc_or(ent3-$_2$) = k2-$_2$))
$\wedge$ ($\forall$ k3-$_1$, k3-$_2$. mk-determine(k3-$_1$, k3-$_2$) in-determine rel$_3$
$\rightarrow$ ($\exists$ ent2-$_1$,ent4-$_2$.ent2-$_1$ in-doctor sent$_2$ $\wedge$ ent4-$_2$ in-cc_data sent$_4$
$\wedge$ key-doctor(ent2-$_1$) = k3-$_1$ $\wedge$ key-cc_data(ent4-$_2$) = k3-$_2$))
$\wedge$ ($\forall$ k4-$_1$, k4-$_2$. mk-make(k4-$_1$, k4-$_2$) in-make rel$_4$
$\rightarrow$ ($\exists$ ent2-$_1$,ent5-$_2$.ent2-$_1$ in-doctor sent$_2$ $\wedge$ ent5-$_2$ in-cc_findings sent$_5$
$\wedge$ key-doctor(ent2-$_1$) = k4-$_1$ $\wedge$ key-cc_findings(ent5-$_2$) = k4-$_2$))
$\wedge$ ($\forall$ k5-$_1$, k5-$_2$. mk-examination(k5-$_1$, k5-$_2$) in-examination rel$_5$
$\rightarrow$ ($\exists$ ent3-$_1$,ent4-$_2$.ent3-$_1$ in-cc_or sent$_3$ $\wedge$ ent4-$_2$ in-cc_data sent$_4$
$\wedge$ key-cc_or(ent3-$_1$) = k5-$_1$ $\wedge$ key-cc_data(ent4-$_2$) = k5-$_2$))
$\wedge$ ($\forall$ k6-$_1$, k6-$_2$. mk-finding(k6-$_1$, k6-$_2$) in-finding rel$_6$
$\rightarrow$ ($\exists$ ent4-$_1$,ent5-$_2$.ent4-$_1$ in-cc_data sent$_4$ $\wedge$ ent5-$_2$ in-cc_findings sent$_5$
$\wedge$ key-cc_data(ent4-$_1$) = k6-$_1$ $\wedge$ key-cc_findings(ent5-$_2$) = k6-$_2$))
$\wedge$ sent$_1$ $\neq$ errorset-patient $\wedge$ sent$_2$ $\neq$ errorset-doctor $\wedge$ sent$_3$ $\neq$ errorset-cc_or
$\wedge$ sent$_4$ $\neq$ errorset-cc_data $\wedge$ sent$_5$ $\neq$ errorset-cc_findings,

$\text{mk-db}(\text{sent}_1, \text{sent}_2, \text{sent}_3, \text{sent}_4, \text{sent}_5, \text{rel}_1, \text{rel}_2, \text{rel}_3, \text{rel}_4, \text{rel}_5, \text{rel}_6) \neq \text{error-db}$
$\rightarrow (\text{mk-db}(\text{sent}_1, \text{sent}_2, \text{sent}_3, \text{sent}_4, \text{sent}_5, \text{rel}_1, \text{rel}_2, \text{rel}_3, \text{rel}_4, \text{rel}_5, \text{rel}_6)$
$\quad = \text{mk-db}(\text{sent1-}_1, \text{sent2-}_1, \text{sent3-}_1, \text{sent4-}_1, \text{sent5-}_1,$
$\qquad \text{rel1-}_1, \text{rel2-}_1, \text{rel3-}_1, \text{rel4-}_1, \text{rel5-}_1, \text{rel6-}_1)$
$\quad \rightarrow \text{sent}_1 = \text{sent1-}_1 \wedge \text{sent}_2 = \text{sent2-}_1 \wedge \text{sent}_3 = \text{sent3-}_1$
$\qquad \wedge\ \text{sent}_4 = \text{sent4-}_1 \wedge \text{sent}_5 = \text{sent5-}_1$
$\qquad \wedge\ \text{rel}_1 = \text{rel1-}_1 \wedge \text{rel}_2 = \text{rel2-}_1 \wedge \text{rel}_3 = \text{rel3-}_1$
$\qquad \wedge\ \text{rel}_4 = \text{rel4-}_1 \wedge \text{rel}_5 = \text{rel5-}_1 \wedge \text{rel}_6 = \text{rel6-}_1),$
$\text{empty-db} = \text{mk-db}(\text{emptyset-patient}, \text{emptyset-doctor}, \text{emptyset-cc\_or},$
$\qquad \text{emptyset-cc\_data}, \text{emptyset-cc\_findings},$
$\qquad \text{emptyrel-part\_of}, \text{emptyrel-orders}, \text{emptyrel-determine},$
$\qquad \text{emptyrel-make}, \text{emptyrel-examination}, \text{emptyrel-finding}),$
$\text{vdb} \neq \text{error-db}$
$\wedge\ \text{vdb} = \text{mk-db}(\text{sent}_1, \text{sent}_2, \text{sent}_3, \text{sent}_4, \text{sent}_5, \text{rel}_1, \text{rel}_2, \text{rel}_3, \text{rel}_4, \text{rel}_5, \text{rel}_6)$
$\rightarrow \text{ent-patient}(\text{vdb}) = \text{sent}_1 \wedge \text{ent-doctor}(\text{vdb}) = \text{sent}_2$
$\quad \wedge\ \text{ent-cc\_or}(\text{vdb}) = \text{sent}_3 \wedge \text{ent-cc\_data}(\text{vdb}) = \text{sent}_4$
$\quad \wedge\ \text{ent-cc\_findings}(\text{vdb}) = \text{sent}_5$
$\quad \wedge\ \text{put-patient}(\text{ent}_1, \text{vdb})$
$\quad = \text{mk-db}(\text{sent}_1 +_{patient} \text{ent}_1,$
$\qquad \text{sent}_2, \text{sent}_3, \text{sent}_4, \text{sent}_5, \text{rel}_1, \text{rel}_2, \text{rel}_3, \text{rel}_4, \text{rel}_5, \text{rel}_6)$
$\quad \wedge\ \text{put-doctor}(\text{ent}_2, \text{vdb})$
$\quad = \text{mk-db}(\text{sent}_1,$
$\qquad \text{sent}_2 +_{doctor} \text{ent}_2,$
$\qquad \text{sent}_3, \text{sent}_4, \text{sent}_5, \text{rel}_1, \text{rel}_2, \text{rel}_3, \text{rel}_4, \text{rel}_5, \text{rel}_6)$
$\quad \wedge\ \text{put-cc\_or}(\text{ent}_3, \text{vdb})$
$\quad = \text{mk-db}(\text{sent}_1, \text{sent}_2,$
$\qquad \text{sent}_3 +_{cc\_or} \text{ent}_3,$
$\qquad \text{sent}_4, \text{sent}_5, \text{rel}_1, \text{rel}_2, \text{rel}_3, \text{rel}_4, \text{rel}_5, \text{rel}_6)$
$\quad \wedge\ \text{put-cc\_data}(\text{ent}_4, \text{vdb})$
$\quad = \text{mk-db}(\text{sent}_1, \text{sent}_2, \text{sent}_3,$
$\qquad \text{sent}_4 +_{cc\_data} \text{ent}_4,$
$\qquad \text{sent}_5, \text{rel}_1, \text{rel}_2, \text{rel}_3, \text{rel}_4, \text{rel}_5, \text{rel}_6)$
$\quad \wedge\ \text{put-cc\_findings}(\text{ent}_5, \text{vdb})$
$\quad = \text{mk-db}(\text{sent}_1, \text{sent}_2, \text{sent}_3, \text{sent}_4,$
$\qquad \text{sent}_5 +_{cc\_findings} \text{ent}_5,$
$\qquad \text{rel}_1, \text{rel}_2, \text{rel}_3, \text{rel}_4, \text{rel}_5, \text{rel}_6)$
$\quad \wedge\ \text{del-patient}(\text{ent}_1, \text{vdb})$
$\quad = \text{mk-db}(\text{sent}_1 -_{patient} \text{ent}_1,$
$\qquad \text{sent}_2, \text{sent}_3, \text{sent}_4, \text{sent}_5, \text{rel}_1, \text{rel}_2, \text{rel}_3, \text{rel}_4, \text{rel}_5, \text{rel}_6)$
$\quad \wedge\ \text{del-doctor}(\text{ent}_2, \text{vdb})$
$\quad = \text{mk-db}(\text{sent}_1,$
$\qquad \text{sent}_2 -_{doctor} \text{ent}_2,$
$\qquad \text{sent}_3, \text{sent}_4, \text{sent}_5, \text{rel}_1, \text{rel}_2, \text{rel}_3, \text{rel}_4, \text{rel}_5, \text{rel}_6)$
$\quad \wedge\ \text{del-cc\_or}(\text{ent}_3, \text{vdb})$
$\quad = \text{mk-db}(\text{sent}_1, \text{sent}_2,$
$\qquad \text{sent}_3 -_{cc\_or} \text{ent}_3,$
$\qquad \text{sent}_4, \text{sent}_5, \text{rel}_1, \text{rel}_2, \text{rel}_3, \text{rel}_4, \text{rel}_5, \text{rel}_6)$
$\quad \wedge\ \text{del-cc\_data}(\text{ent}_4, \text{vdb})$
$\quad = \text{mk-db}(\text{sent}_1, \text{sent}_2, \text{sent}_3,$
$\qquad \text{sent}_4 -_{cc\_data} \text{ent}_4,$
$\qquad \text{sent}_5, \text{rel}_1, \text{rel}_2, \text{rel}_3, \text{rel}_4, \text{rel}_5, \text{rel}_6)$
$\quad \wedge\ \text{del-cc\_findings}(\text{ent}_5, \text{vdb})$
$\quad = \text{mk-db}(\text{sent}_1, \text{sent}_2, \text{sent}_3, \text{sent}_4,$
$\qquad \text{sent}_5 -_{cc\_findings} \text{ent}_5,$
$\qquad \text{rel}_1, \text{rel}_2, \text{rel}_3, \text{rel}_4, \text{rel}_5, \text{rel}_6),$
$\text{get-patient}(\text{key}_1, \text{vdb}) \neq \text{error-patient}$

$\leftrightarrow (\exists\ \text{ent}_1.\text{ent}_1\ \text{in-patient ent-patient(vdb)} \wedge \text{key-patient}(\text{ent}_1) = \text{key}_1),$

$\text{get-doctor}(\text{key}_2, \text{vdb}) \neq \text{error-doctor}$

$\leftrightarrow (\exists\ \text{ent}_2.\text{ent}_2\ \text{in-doctor ent-doctor(vdb)} \wedge \text{key-doctor}(\text{ent}_2) = \text{key}_2),$

$\text{get-cc\_or}(\text{key}_3, \text{vdb}) \neq \text{error-cc\_or}$

$\leftrightarrow (\exists\ \text{ent}_3.\text{ent}_3\ \text{in-cc\_or ent-cc\_or(vdb)} \wedge \text{key-cc\_or}(\text{ent}_3) = \text{key}_3),$

$\text{get-cc\_data}(\text{key}_4, \text{vdb}) \neq \text{error-cc\_data}$

$\leftrightarrow (\exists\ \text{ent}_4.\text{ent}_4\ \text{in-cc\_data ent-cc\_data(vdb)} \wedge \text{key-cc\_data}(\text{ent}_4) = \text{key}_4),$

$\text{get-cc\_findings}(\text{key}_5, \text{vdb}) \neq \text{error-cc\_findings}$

$\leftrightarrow (\exists\ \text{ent}_5.\text{ent}_5\ \text{in-cc\_findings ent-cc\_findings(vdb)} \wedge \text{key-cc\_findings}(\text{ent}_5) = \text{key}_5),$

$\text{ent}_1 \neq \text{error-patient}$

$\rightarrow (\text{get-patient}(\text{key}_1, \text{vdb}) = \text{ent}_1$

$\quad \leftrightarrow \text{ent}_1\ \text{in-patient ent-patient(vdb)} \wedge \text{key-patient}(\text{ent}_1) = \text{key}_1),$

$\text{ent}_2 \neq \text{error-doctor}$

$\rightarrow (\text{get-doctor}(\text{key}_2, \text{vdb}) = \text{ent}_2$

$\quad \leftrightarrow \text{ent}_2\ \text{in-doctor ent-doctor(vdb)} \wedge \text{key-doctor}(\text{ent}_2) = \text{key}_2),$

$\text{ent}_3 \neq \text{error-cc\_or}$

$\rightarrow (\text{get-cc\_or}(\text{key}_3, \text{vdb}) = \text{ent}_3$

$\quad \leftrightarrow \text{ent}_3\ \text{in-cc\_or ent-cc\_or(vdb)} \wedge \text{key-cc\_or}(\text{ent}_3) = \text{key}_3),$

$\text{ent}_4 \neq \text{error-cc\_data}$

$\rightarrow (\text{get-cc\_data}(\text{key}_4, \text{vdb}) = \text{ent}_4$

$\quad \leftrightarrow \text{ent}_4\ \text{in-cc\_data ent-cc\_data(vdb)} \wedge \text{key-cc\_data}(\text{ent}_4) = \text{key}_4),$

$\text{ent}_5 \neq \text{error-cc\_findings}$

$\rightarrow (\text{get-cc\_findings}(\text{key}_5, \text{vdb}) = \text{ent}_5$

$\quad \leftrightarrow \text{ent}_5\ \text{in-cc\_findings ent-cc\_findings(vdb)} \wedge \text{key-cc\_findings}(\text{ent}_5) = \text{key}_5),$

$\text{update-patient}(\text{key}_1, \text{ent}_1, \text{vdb}) \neq \text{error-db}$

$\leftrightarrow (\exists\ \text{ent1-}_2.\text{ent1-}_2\ \text{in-patient ent-patient(vdb)}$

$\qquad\qquad \wedge\ \text{key-patient}(\text{ent1-}_2) = \text{key}_1$

$\qquad\qquad \wedge\ \text{key-patient}(\text{ent}_1) = \text{key}_1),$

$\text{update-doctor}(\text{key}_2, \text{ent}_2, \text{vdb}) \neq \text{error-db}$

$\leftrightarrow (\exists\ \text{ent2-}_2.\text{ent2-}_2\ \text{in-doctor ent-doctor(vdb)}$

$\qquad\qquad \wedge\ \text{key-doctor}(\text{ent2-}_2) = \text{key}_2$

$\qquad\qquad \wedge\ \text{key-doctor}(\text{ent}_2) = \text{key}_2),$

$\text{update-cc\_or}(\text{key}_3, \text{ent}_3, \text{vdb}) \neq \text{error-db}$

$\leftrightarrow (\exists\ \text{ent3-}_2.\text{ent3-}_2\ \text{in-cc\_or ent-cc\_or(vdb)}$

$\qquad\qquad \wedge\ \text{key-cc\_or}(\text{ent3-}_2) = \text{key}_3$

$\qquad\qquad \wedge\ \text{key-cc\_or}(\text{ent}_3) = \text{key}_3),$

$\text{update-cc\_data}(\text{key}_4, \text{ent}_4, \text{vdb}) \neq \text{error-db}$

$\leftrightarrow (\exists\ \text{ent4-}_2.\text{ent4-}_2\ \text{in-cc\_data ent-cc\_data(vdb)}$

$\qquad\qquad \wedge\ \text{key-cc\_data}(\text{ent4-}_2) = \text{key}_4$

$\qquad\qquad \wedge\ \text{key-cc\_data}(\text{ent}_4) = \text{key}_4),$

$\text{update-cc\_findings}(\text{key}_5, \text{ent}_5, \text{vdb}) \neq \text{error-db}$

$\leftrightarrow (\exists\ \text{ent5-}_2.\text{ent5-}_2\ \text{in-cc\_findings ent-cc\_findings(vdb)}$

$\qquad\qquad \wedge\ \text{key-cc\_findings}(\text{ent5-}_2) = \text{key}_5$

$\qquad\qquad \wedge\ \text{key-cc\_findings}(\text{ent}_5) = \text{key}_5),$

$\text{update-patient}(\text{key}_1, \text{ent}_1, \text{vdb}) \neq \text{error-db}$

$\wedge\ \text{vdb} = \text{mk-db}(\text{sent}_1, \text{sent}_2, \text{sent}_3, \text{sent}_4, \text{sent}_5, \text{rel}_1, \text{rel}_2, \text{rel}_3, \text{rel}_4, \text{rel}_5, \text{rel}_6)$

$\rightarrow \text{update-patient}(\text{key}_1, \text{ent}_1, \text{vdb})$

$\quad = \text{mk-db}(\text{sent}_1 -_{patient} \text{get-patient}(\text{key}_1, \text{vdb}) +_{patient} \text{ent}_1,$

$\qquad\qquad \text{sent}_2, \text{sent}_3, \text{sent}_4, \text{sent}_5, \text{rel}_1, \text{rel}_2, \text{rel}_3, \text{rel}_4, \text{rel}_5, \text{rel}_6),$

$\text{update-doctor}(\text{key}_2, \text{ent}_2, \text{vdb}) \neq \text{error-db}$

$\wedge\ \text{vdb} = \text{mk-db}(\text{sent}_1, \text{sent}_2, \text{sent}_3, \text{sent}_4, \text{sent}_5, \text{rel}_1, \text{rel}_2, \text{rel}_3, \text{rel}_4, \text{rel}_5, \text{rel}_6)$

$\rightarrow \text{update-doctor}(\text{key}_2, \text{ent}_2, \text{vdb})$

$\quad = \text{mk-db}(\text{sent}_1,$

$\qquad\qquad \text{sent}_2 -_{doctor} \text{get-doctor}(\text{key}_2, \text{vdb}) +_{doctor} \text{ent}_2,$

$\qquad\qquad \text{sent}_3, \text{sent}_4, \text{sent}_5, \text{rel}_1, \text{rel}_2, \text{rel}_3, \text{rel}_4, \text{rel}_5, \text{rel}_6),$

$\text{update-cc\_or}(\text{key}_3, \text{ent}_3, \text{vdb}) \neq \text{error-db}$

$\wedge$ vdb = mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$, rel$_2$, rel$_3$, rel$_4$, rel$_5$, rel$_6$)
$\rightarrow$ update-cc_or(key$_3$, ent$_3$, vdb)
  = mk-db(sent$_1$, sent$_2$,
    sent$_3$ -$_{cc\_or}$ get-cc_or(key$_3$, vdb) +$_{cc\_or}$ ent$_3$,
    sent$_4$, sent$_5$, rel$_1$, rel$_2$, rel$_3$, rel$_4$, rel$_5$, rel$_6$),
update-cc_data(key$_4$, ent$_4$, vdb) $\neq$ error-db
$\wedge$ vdb = mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$, rel$_2$, rel$_3$, rel$_4$, rel$_5$, rel$_6$)
$\rightarrow$ update-cc_data(key$_4$, ent$_4$, vdb)
  = mk-db(sent$_1$, sent$_2$, sent$_3$,
    sent$_4$ -$_{cc\_data}$ get-cc_data(key$_4$, vdb) +$_{cc\_data}$ ent$_4$,
    sent$_5$, rel$_1$, rel$_2$, rel$_3$, rel$_4$, rel$_5$, rel$_6$),
update-cc_findings(key$_5$, ent$_5$, vdb) $\neq$ error-db
$\wedge$ vdb = mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$, rel$_2$, rel$_3$, rel$_4$, rel$_5$, rel$_6$)
$\rightarrow$ update-cc_findings(key$_5$, ent$_5$, vdb)
  = mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$,
    sent$_5$ -$_{cc\_findings}$ get-cc_findings(key$_5$, vdb) +$_{cc\_findings}$ ent$_5$,
    rel$_1$, rel$_2$, rel$_3$, rel$_4$, rel$_5$, rel$_6$),
vdb $\neq$ error-db
$\wedge$ vdb = mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$, rel$_2$, rel$_3$, rel$_4$, rel$_5$, rel$_6$)
$\wedge$ ent1-$_1$ $\neq$ error-patient
$\wedge$ ent3-$_2$ $\neq$ error-cc_or
$\rightarrow$ (part_of(vdb, ent1-$_1$, ent3-$_2$)
$\leftrightarrow$
mk-part_of(key-patient(ent1-$_1$), key-cc_or(ent3-$_2$)) in-part_of rel$_1$)
$\wedge$ est-part_of(vdb, ent1-$_1$, ent3-$_2$)
  = mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$,
    rel$_1$ +$_{part\_of}$ mk-part_of(key-patient(ent1-$_1$),
      key-cc_or(ent3-$_2$)),
    rel$_2$, rel$_3$, rel$_4$, rel$_5$, rel$_6$)
$\wedge$ rel-part_of(vdb, ent1-$_1$, ent3-$_2$)
  = mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$,
    rel$_1$ -$_{part\_of}$ mk-part_of(key-patient(ent1-$_1$),
      key-cc_or(ent3-$_2$)),
    rel$_2$, rel$_3$, rel$_4$, rel$_5$, rel$_6$),
vdb $\neq$ error-db
$\wedge$ vdb = mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$, rel$_2$, rel$_3$, rel$_4$, rel$_5$, rel$_6$)
$\wedge$ ent2-$_1$ $\neq$ error-doctor
$\wedge$ ent3-$_2$ $\neq$ error-cc_or
$\rightarrow$ (orders(vdb, ent2-$_1$, ent3-$_2$)
$\leftrightarrow$
mk-orders(key-doctor(ent2-$_1$), key-cc_or(ent3-$_2$)) in-orders rel$_2$)
$\wedge$ est-orders(vdb, ent2-$_1$, ent3-$_2$)
  = mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$,
    rel$_2$ +$_{orders}$ mk-orders(key-doctor(ent2-$_1$),
      key-cc_or(ent3-$_2$)),
    rel$_3$, rel$_4$, rel$_5$, rel$_6$)
$\wedge$ rel-orders(vdb, ent2-$_1$, ent3-$_2$)
  = mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$,
    rel$_2$ -$_{orders}$ mk-orders(key-doctor(ent2-$_1$),
      key-cc_or(ent3-$_2$)),
    rel$_3$, rel$_4$, rel$_5$, rel$_6$),
vdb $\neq$ error-db
$\wedge$ vdb = mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$, rel$_2$, rel$_3$, rel$_4$, rel$_5$, rel$_6$)
$\wedge$ ent2-$_1$ $\neq$ error-doctor
$\wedge$ ent4-$_2$ $\neq$ error-cc_data
$\rightarrow$ (determine(vdb, ent2-$_1$, ent4-$_2$)

$\leftrightarrow$
mk-determine(key-doctor(ent2-$_1$), key-cc_data(ent4-$_2$)) in-determine rel$_3$)
$\wedge$ est-determine(vdb, ent2-$_1$, ent4-$_2$)
= mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$, rel$_2$,
rel$_3$ $+_{determine}$ mk-determine(key-doctor(ent2-$_1$),
key-cc_data(ent4-$_2$)),
rel$_4$, rel$_5$, rel$_6$)
$\wedge$ rel-determine(vdb, ent2-$_1$, ent4-$_2$)
= mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$, rel$_2$,
rel$_3$ $-_{determine}$ mk-determine(key-doctor(ent2-$_1$),
key-cc_data(ent4-$_2$)),
rel$_4$, rel$_5$, rel$_6$),
vdb $\neq$ error-db
$\wedge$ vdb = mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$, rel$_2$, rel$_3$, rel$_4$, rel$_5$, rel$_6$)
$\wedge$ ent2-$_1$ $\neq$ error-doctor
$\wedge$ ent5-$_2$ $\neq$ error-cc_findings
$\rightarrow$ (make(vdb, ent2-$_1$, ent5-$_2$)
$\leftrightarrow$
mk-make(key-doctor(ent2-$_1$), key-cc_findings(ent5-$_2$)) in-make rel$_4$)
$\wedge$ est-make(vdb, ent2-$_1$, ent5-$_2$)
= mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$, rel$_2$, rel$_3$,
rel$_4$ $+_{make}$ mk-make(key-doctor(ent2-$_1$),
key-cc_findings(ent5-$_2$)),
rel$_5$, rel$_6$)
$\wedge$ rel-make(vdb, ent2-$_1$, ent5-$_2$)
= mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$, rel$_2$, rel$_3$,
rel$_4$ $-_{make}$ mk-make(key-doctor(ent2-$_1$),
key-cc_findings(ent5-$_2$)),
rel$_5$, rel$_6$),
vdb $\neq$ error-db
$\wedge$ vdb = mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$, rel$_2$, rel$_3$, rel$_4$, rel$_5$, rel$_6$)
$\wedge$ ent3-$_1$ $\neq$ error-cc_or
$\wedge$ ent4-$_2$ $\neq$ error-cc_data
$\rightarrow$ (examination(vdb, ent3-$_1$, ent4-$_2$)
$\leftrightarrow$
mk-examination(key-cc_or(ent3-$_1$), key-cc_data(ent4-$_2$)) in-examination rel$_5$)
$\wedge$ est-examination(vdb, ent3-$_1$, ent4-$_2$)
= mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$, rel$_2$, rel$_3$, rel$_4$,
rel$_5$ $+_{examination}$ mk-examination(key-cc_or(ent3-$_1$),
key-cc_data(ent4-$_2$)),
rel$_6$)
$\wedge$ rel-examination(vdb, ent3-$_1$, ent4-$_2$)
= mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$, rel$_2$, rel$_3$, rel$_4$,
rel$_5$ $-_{examination}$ mk-examination(key-cc_or(ent3-$_1$),
key-cc_data(ent4-$_2$)),
rel$_6$),
vdb $\neq$ error-db
$\wedge$ vdb = mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$, rel$_2$, rel$_3$, rel$_4$, rel$_5$, rel$_6$)
$\wedge$ ent4-$_1$ $\neq$ error-cc_data
$\wedge$ ent5-$_2$ $\neq$ error-cc_findings
$\rightarrow$ (finding(vdb, ent4-$_1$, ent5-$_2$)
$\leftrightarrow$
mk-finding(key-cc_data(ent4-$_1$), key-cc_findings(ent5-$_2$)) in-finding rel$_6$)
$\wedge$ est-finding(vdb, ent4-$_1$, ent5-$_2$)
= mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$, rel$_2$, rel$_3$, rel$_4$, rel$_5$,
rel$_6$ $+_{finding}$ mk-finding(key-cc_data(ent4-$_1$),

key-cc_findings(ent5-$_2$)))
∧ rel-finding(vdb, ent4-$_1$, ent5-$_2$)
= mk-db(sent$_1$, sent$_2$, sent$_3$, sent$_4$, sent$_5$, rel$_1$, rel$_2$, rel$_3$, rel$_4$, rel$_5$,
rel$_6$ -$_{finding}$ mk-finding(key-cc_data(ent4-$_1$),
key-cc_findings(ent5-$_2$))),
ent-patient(error-db) = errorset-patient,
ent-doctor(error-db) = errorset-doctor,
ent-cc_or(error-db) = errorset-cc_or,
ent-cc_data(error-db) = errorset-cc_data,
ent-cc_findings(error-db) = errorset-cc_findings,
put-patient(ent$_1$, error-db) = error-db,
put-doctor(ent$_2$, error-db) = error-db,
put-cc_or(ent$_3$, error-db) = error-db,
put-cc_data(ent$_4$, error-db) = error-db,
put-cc_findings(ent$_5$, error-db) = error-db,
del-patient(ent$_1$, error-db) = error-db,
del-doctor(ent$_2$, error-db) = error-db,
del-cc_or(ent$_3$, error-db) = error-db,
del-cc_data(ent$_4$, error-db) = error-db,
del-cc_findings(ent$_5$, error-db) = error-db,
vdb = error-db ∨ ent1-$_1$ = error-patient ∨ ent3-$_2$ = error-cc_or
→ ¬ part_of(vdb, ent1-$_1$, ent3-$_2$)
∧ est-part_of(vdb, ent1-$_1$, ent3-$_2$) = error-db
∧ rel-part_of(vdb, ent1-$_1$, ent3-$_2$) = error-db,
vdb = error-db ∨ ent2-$_1$ = error-doctor ∨ ent3-$_2$ = error-cc_or
→ ¬ orders(vdb, ent2-$_1$, ent3-$_2$)
∧ est-orders(vdb, ent2-$_1$, ent3-$_2$) = error-db
∧ rel-orders(vdb, ent2-$_1$, ent3-$_2$) = error-db,
vdb = error-db ∨ ent2-$_1$ = error-doctor ∨ ent4-$_2$ = error-cc_data
→ ¬ determine(vdb, ent2-$_1$, ent4-$_2$)
∧ est-determine(vdb, ent2-$_1$, ent4-$_2$) = error-db
∧ rel-determine(vdb, ent2-$_1$, ent4-$_2$) = error-db,
vdb = error-db ∨ ent2-$_1$ = error-doctor ∨ ent5-$_2$ = error-cc_findings
→ ¬ make(vdb, ent2-$_1$, ent5-$_2$)
∧ est-make(vdb, ent2-$_1$, ent5-$_2$) = error-db
∧ rel-make(vdb, ent2-$_1$, ent5-$_2$) = error-db,
vdb = error-db ∨ ent3-$_1$ = error-cc_or ∨ ent4-$_2$ = error-cc_data
→ ¬ examination(vdb, ent3-$_1$, ent4-$_2$)
∧ est-examination(vdb, ent3-$_1$, ent4-$_2$) = error-db
∧ rel-examination(vdb, ent3-$_1$, ent4-$_2$) = error-db,
vdb = error-db ∨ ent4-$_1$ = error-cc_data ∨ ent5-$_2$ = error-cc_findings
→ ¬ finding(vdb, ent4-$_1$, ent5-$_2$)
∧ est-finding(vdb, ent4-$_1$, ent5-$_2$) = error-db
∧ rel-finding(vdb, ent4-$_1$, ent5-$_2$) = error-db
end enrich

A.5.2 The Implementation

DB-preDB =
module
export ER_DB_Cardiac-Catheterisation
refinement
representation of sorts
pre-db implements db;
representation of operations

```
    empty-db#     implements   empty-db;
    error-db#     implements   error-db;
    mk-db#        implements   mk-db;
    ent-patient#             implements   ent-patient;
    ent-doctor#              implements   ent-doctor;
    ent-cc_or#               implements   ent-cc_or;
    ent-cc_data#             implements   ent-cc_data;
    ent-cc_findings#         implements   ent-cc_findings;
    put-patient#             implements   put-patient;
    put-doctor#              implements   put-doctor;
    put-cc_or#               implements   put-cc_or;
    put-cc_data#             implements   put-cc_data;
    put-cc_findings#         implements   put-cc_findings;
    del-patient#             implements   del-patient;
    del-doctor#              implements   del-doctor;
    del-cc_or#               implements   del-cc_or;
    del-cc_data#             implements   del-cc_data;
    del-cc_findings#         implements   del-cc_findings;
    get-patient#             implements   get-patient;
    get-doctor#              implements   get-doctor;
    get-cc_or#               implements   get-cc_or;
    get-cc_data#             implements   get-cc_data;
    get-cc_findings#         implements   get-cc_findings;
    update-patient#          implements   update-patient;
    update-doctor#           implements   update-doctor;
    update-cc_or#            implements   update-cc_or;
    update-cc_data#          implements   update-cc_data;
    update-cc_findings#      implements   update-cc_findings;
    est-part_of#             implements   est-part_of;
    est-orders#              implements   est-orders;
    est-determine#           implements   est-determine;
    est-make#                implements   est-make;
    est-examination#         implements   est-examination;
    est-finding#             implements   est-finding;
    rel-part_of#             implements   rel-part_of;
    rel-orders#              implements   rel-orders;
    rel-determine#           implements   rel-determine;
    rel-make#                implements   rel-make;
    rel-examination#         implements   rel-examination;
    rel-finding#             implements   rel-finding;
    part_of#                 implements   part_of;
    orders#                  implements   orders;
    determine#               implements   determine;
    make#                    implements   make;
    examination#             implements   examination;
    finding#                 implements   finding;

import pre-DB_Cardiac-Catheterisation

procedures
     empty-db#    ()   : pre-db;
     error-db#    ()   : pre-db;
```

mk-db#	(set-patient, set-doctor, set-cc_or, set-cc_data, set-cc_findings, reltype-part_of, reltype-orders, reltype-determine, reltype-make, reltype-examination, reltype-finding)	: pre-db;
ent-patient#	(pre-db)	: set-patient;
ent-doctor#	(pre-db)	: set-doctor;
ent-cc_or#	(pre-db)	: set-cc_or;
ent-cc_data#	(pre-db)	: set-cc_data;
ent-cc_findings#	(pre-db)	: set-cc_findings;
put-patient#	(patient, pre-db)	: pre-db;
put-doctor#	(doctor, pre-db)	: pre-db;
put-cc_or#	(cc_or, pre-db)	: pre-db;
put-cc_data#	(cc_data, pre-db)	: pre-db;
put-cc_findings#	(cc_findings, pre-db)	: pre-db;
del-patient#	(patient, pre-db)	: pre-db;
del-doctor#	(doctor, pre-db)	: pre-db;
del-cc_or#	(cc_or, pre-db)	: pre-db;
del-cc_data#	(cc_data, pre-db)	: pre-db;
del-cc_findings#	(cc_findings, pre-db)	: pre-db;
get-patient#	(keysort-patient, pre-db)	: patient;
get-doctor#	(keysort-doctor, pre-db)	: doctor;
get-cc_or#	(keysort-cc_or, pre-db)	: cc_or;
get-cc_data#	(keysort-cc_data, pre-db)	: cc_data;
get-cc_findings#	(keysort-cc_findings, pre-db)	: cc_findings;
update-patient#	(keysort-patient, patient, pre-db)	: pre-db;
update-doctor#	(keysort-doctor, doctor, pre-db)	: pre-db;
update-cc_or#	(keysort-cc_or, cc_or, pre-db)	: pre-db;
update-cc_data#	(keysort-cc_data, cc_data, pre-db)	: pre-db;
update-cc_findings#	(keysort-cc_findings, cc_findings, pre-db)	: pre-db;
est-part_of#	(pre-db, patient, cc_or)	: pre-db;
est-orders#	(pre-db, doctor, cc_or)	: pre-db;
est-determine#	(pre-db, doctor, cc_data)	: pre-db;
est-make#	(pre-db, doctor, cc_findings)	: pre-db;
est-examination#	(pre-db, cc_or, cc_data)	: pre-db;
est-finding#	(pre-db, cc_data, cc_findings)	: pre-db;
rel-part_of#	(pre-db, patient, cc_or)	: pre-db;
rel-orders#	(pre-db, doctor, cc_or)	: pre-db;
rel-determine#	(pre-db, doctor, cc_data)	: pre-db;
rel-make#	(pre-db, doctor, cc_findings)	: pre-db;
rel-examination#	(pre-db, cc_or, cc_data)	: pre-db;
rel-finding#	(pre-db, cc_data, cc_findings)	: pre-db;
part_of#	(pre-db, patient, cc_or)	: bool;
orders#	(pre-db, doctor, cc_or)	: bool;
determine#	(pre-db, doctor, cc_data)	: bool;
make#	(pre-db, doctor, cc_findings)	: bool;
examination#	(pre-db, cc_or, cc_data)	: bool;
finding#	(pre-db, cc_data, cc_findings)	: bool;
legal-part_of#	(reltype-part_of, set-patient, set-cc_or)	: bool;
legal-orders#	(reltype-orders, set-doctor, set-cc_or)	: bool;
legal-determine#	(reltype-determine, set-doctor, set-cc_data)	: bool;
legal-make#	(reltype-make, set-doctor, set-cc_findings)	: bool;

```
legal-examination#    (reltype-examination, set-cc_or,
                      set-cc_data)                              : bool;
legal-finding#        (reltype-finding, set-cc_data,
                      set-cc_findings)                          : bool;
in-fst-part_of#       (keysort-patient, reltype-part_of)        : bool;
in-fst-orders#        (keysort-doctor, reltype-orders)          : bool;
in-fst-determine#     (keysort-doctor, reltype-determine)       : bool;
in-fst-make#          (keysort-doctor, reltype-make)            : bool;
in-fst-examination#   (keysort-cc_or, reltype-examination)      : bool;
in-fst-finding#       (keysort-cc_data, reltype-finding)        : bool;
in-snd-part_of#       (keysort-cc_or, reltype-part_of)          : bool;
in-snd-orders#        (keysort-cc_or, reltype-orders)           : bool;
in-snd-determine#     (keysort-cc_data, reltype-determine)      : bool;
in-snd-make#          (keysort-cc_findings, reltype-make)       : bool;
in-snd-examination#   (keysort-cc_data, reltype-examination)    : bool;
in-snd-finding#       (keysort-cc_findings, reltype-finding)    : bool;
```

variables pdb_1, pdb: pre-db; b: bool;

implementation

```
empty-db#(var pdb)
begin
  pdb := p-mk-db(emptyset-patient,
                 emptyset-doctor,
                 emptyset-cc_or,
                 emptyset-cc_data,
                 emptyset-cc_findings,
                 emptyrel-part_of,
                 emptyrel-orders,
                 emptyrel-determine,
                 emptyrel-make,
                 emptyrel-examination,
                 emptyrel-finding)
end
```

```
error-db#(var pdb) begin pdb := p-error-db end
```

```
mk-db#(sent1, sent2, sent3, sent4, sent5, rel1, rel2, rel3, rel4, rel5, rel6; var pdb)
begin
  if sent1 = errorset-patient
    ∨ sent2 = errorset-doctor
    ∨ sent3 = errorset-cc_or
    ∨ sent4 = errorset-cc_data
    ∨ sent5 = errorset-cc_findings then
   pdb := p-error-db
  else
   var b = tt in
   begin
     legal-part_of#(rel1, sent1, sent3;b);
     if b = ff then pdb := p-error-db else
```

```
          begin
             legal-orders#(rel2, sent2, sent3;b);
             if b = ff then pdb := p-error-db else
               begin
                  legal-determine#(rel3, sent2, sent4;b);
                  if b = ff then pdb := p-error-db else
                    begin
                       legal-make#(rel4, sent2, sent5;b);
                       if b = ff then pdb := p-error-db else
                         begin
                            legal-examination#(rel5, sent3, sent4;b);
                            if b = ff then pdb := p-error-db else
                              begin
                                 legal-finding#(rel6, sent4, sent5;b);
                                 if b = ff then pdb := p-error-db else
                                   pdb := p-mk-db(sent1, sent2, sent3, sent4, sent5,
                                                  rel1, rel2, rel3, rel4, rel5, rel6)
                              end
                         end
                    end
               end
          end
     end
end
```

```
ent-patient#(pdb; var sent1)
begin
   if pdb = p-error-db then sent1 := errorset-patient else
     sent1 := p-ent-patient(pdb)
end
```

```
ent-doctor#(pdb; var sent2)
begin
   if pdb = p-error-db then sent2 := errorset-doctor else
     sent2 := p-ent-doctor(pdb)
end
```

```
ent-cc_or#(pdb; var sent3)
begin
   if pdb = p-error-db then sent3 := errorset-cc_or else
     sent3 := p-ent-cc_or(pdb)
end
```

```
ent-cc_data#(pdb; var sent4)
begin
   if pdb = p-error-db then sent4 := errorset-cc_data else
     sent4 := p-ent-cc_data(pdb)
end
```

```
ent-cc_findings#(pdb; var sent5)
begin
   if pdb = p-error-db then sent5 := errorset-cc_findings else
     sent5 := p-ent-cc_findings(pdb)
end


put-patient#(ent1, pdb; var pdb1)
begin
   if pdb = p-error-db then pdb1 := p-error-db else
     var sent1 = p-ent-patient(pdb) +_patient ent1 in
     if sent1 = errorset-patient then pdb1 := p-error-db else
       pdb1 := p-mk-db(sent1,
                       p-ent-doctor(pdb),
                       p-ent-cc_or(pdb),
                       p-ent-cc_data(pdb),
                       p-ent-cc_findings(pdb),
                       p-part_of(pdb),
                       p-orders(pdb),
                       p-determine(pdb),
                       p-make(pdb),
                       p-examination(pdb),
                       p-finding(pdb))
end


put-doctor#(ent2, pdb; var pdb1)
begin
   if pdb = p-error-db then pdb1 := p-error-db else
     var sent2 = p-ent-doctor(pdb) +_doctor ent2 in
     if sent2 = errorset-doctor then pdb1 := p-error-db else
       pdb1 := p-mk-db(p-ent-patient(pdb),
                       sent2,
                       p-ent-cc_or(pdb),
                       p-ent-cc_data(pdb),
                       p-ent-cc_findings(pdb),
                       p-part_of(pdb),
                       p-orders(pdb),
                       p-determine(pdb),
                       p-make(pdb),
                       p-examination(pdb),
                       p-finding(pdb))
end


put-cc_or#(ent3, pdb; var pdb1)
begin
   if pdb = p-error-db then pdb1 := p-error-db else
     var sent3 = p-ent-cc_or(pdb) +_cc_or ent3 in
     if sent3 = errorset-cc_or then pdb1 := p-error-db else
       pdb1 := p-mk-db(p-ent-patient(pdb),
                       p-ent-doctor(pdb),
                       sent3,
                       p-ent-cc_data(pdb),
                       p-ent-cc_findings(pdb),
```

```
                              p-part_of(pdb),
                              p-orders(pdb),
                              p-determine(pdb),
                              p-make(pdb),
                              p-examination(pdb),
                              p-finding(pdb))
end


put-cc_data#(ent4, pdb; var pdb1)
begin
   if pdb = p-error-db then pdb1 := p-error-db else
     var sent4 = p-ent-cc_data(pdb) +cc_data ent4 in
     if sent4 = errorset-cc_data then pdb1 := p-error-db else
       pdb1 := p-mk-db(p-ent-patient(pdb),
                              p-ent-doctor(pdb),
                              p-ent-cc_or(pdb),
                              sent4,
                              p-ent-cc_findings(pdb),
                              p-part_of(pdb),
                              p-orders(pdb),
                              p-determine(pdb),
                              p-make(pdb),
                              p-examination(pdb),
                              p-finding(pdb))
end


put-cc_findings#(ent5, pdb; var pdb1)
begin
   if pdb = p-error-db then pdb1 := p-error-db else
     var sent5 = p-ent-cc_findings(pdb) +cc_findings ent5 in
     if sent5 = errorset-cc_findings then pdb1 := p-error-db else
       pdb1 := p-mk-db(p-ent-patient(pdb),
                              p-ent-doctor(pdb),
                              p-ent-cc_or(pdb),
                              p-ent-cc_data(pdb),
                              sent5,
                              p-part_of(pdb),
                              p-orders(pdb),
                              p-determine(pdb),
                              p-make(pdb),
                              p-examination(pdb),
                              p-finding(pdb))
end


del-patient#(ent1, pdb; var pdb1)
begin
   if ent1 = error-patient ∨ pdb = p-error-db then pdb1 := p-error-db else
     var key1 = key-patient(ent1), b = tt in
     begin
        in-fst-part_of#(key1, p-part_of(pdb);b);
        if b = tt then pdb1 := p-error-db else
          var sent1 = p-ent-patient(pdb) -patient ent1 in
```

```
        pdb_1 := p-mk-db(sent_1,
                         p-ent-doctor(pdb),
                         p-ent-cc_or(pdb),
                         p-ent-cc_data(pdb),
                         p-ent-cc_findings(pdb),
                         p-part_of(pdb),
                         p-orders(pdb),
                         p-determine(pdb),
                         p-make(pdb),
                         p-examination(pdb),
                         p-finding(pdb))
    end
end
```

```
del-doctor#(ent_2, pdb; var pdb_1)
begin
  if ent_2 = error-doctor ∨ pdb = p-error-db then pdb_1 := p-error-db else
    var key_2 = key-doctor(ent_2), b = tt in
    begin
      in-fst-orders#(key_2, p-orders(pdb);b);
      if b = tt then pdb_1 := p-error-db else
        begin
          in-fst-determine#(key_2, p-determine(pdb);b);
          if b = tt then pdb_1 := p-error-db else
            begin
              in-fst-make#(key_2, p-make(pdb);b);
              if b = tt then pdb_1 := p-error-db else
                var sent_2 = p-ent-doctor(pdb) -_doctor ent_2 in
                pdb_1 := p-mk-db(p-ent-patient(pdb),
                                 sent_2,
                                 p-ent-cc_or(pdb),
                                 p-ent-cc_data(pdb),
                                 p-ent-cc_findings(pdb),
                                 p-part_of(pdb),
                                 p-orders(pdb),
                                 p-determine(pdb),
                                 p-make(pdb),
                                 p-examination(pdb),
                                 p-finding(pdb))
            end
        end
    end
end
```

```
del-cc_or#(ent_3, pdb; var pdb_1)
begin
  if ent_3 = error-cc_or ∨ pdb = p-error-db then pdb_1 := p-error-db else
    var key_3 = key-cc_or(ent_3), b = tt in
    begin
      in-fst-examination#(key_3, p-examination(pdb);b);
      if b = tt then pdb_1 := p-error-db else
        begin
          in-snd-part_of#(key_3, p-part_of(pdb);b);
```

```
          if b = tt then pdb_1 := p-error-db else
           begin
             in-snd-orders#(key_3, p-orders(pdb);b);
             if b = tt then pdb_1 := p-error-db else
              var sent_3 = p-ent-cc_or(pdb) -_cc_or ent_3 in
              pdb_1 := p-mk-db(p-ent-patient(pdb),
                               p-ent-doctor(pdb),
                               sent_3,
                               p-ent-cc_data(pdb),
                               p-ent-cc_findings(pdb),
                               p-part_of(pdb),
                               p-orders(pdb),
                               p-determine(pdb),
                               p-make(pdb),
                               p-examination(pdb),
                               p-finding(pdb))
             end
          end
       end
   end
```

```
del-cc_data#(ent_4, pdb; var pdb_1)
begin
   if ent_4 = error-cc_data ∨ pdb = p-error-db then pdb_1 := p-error-db else
    var key_4 = key-cc_data(ent_4), b = tt in
    begin
       in-fst-finding#(key_4, p-finding(pdb);b);
       if b = tt then pdb_1 := p-error-db else
        begin
          in-snd-determine#(key_4, p-determine(pdb);b);
          if b = tt then pdb_1 := p-error-db else
           begin
             in-snd-examination#(key_4, p-examination(pdb);b);
             if b = tt then pdb_1 := p-error-db else
              var sent_4 = p-ent-cc_data(pdb) -_cc_data ent_4 in
              pdb_1 := p-mk-db(p-ent-patient(pdb),
                               p-ent-doctor(pdb),
                               p-ent-cc_or(pdb),
                               sent_4,
                               p-ent-cc_findings(pdb),
                               p-part_of(pdb),
                               p-orders(pdb),
                               p-determine(pdb),
                               p-make(pdb),
                               p-examination(pdb),
                               p-finding(pdb))
           end
        end
    end
end
```

```
del-cc_findings#(ent_5, pdb; var pdb_1)
begin
```

```
  if ent_5 = error-cc_findings ∨ pdb = p-error-db then pdb_1 := p-error-db else
    var key_5 = key-cc_findings(ent_5), b = tt in
    begin
       in-snd-make#(key_5, p-make(pdb);b);
       if b = tt then pdb_1 := p-error-db else
         begin
            in-snd-finding#(key_5, p-finding(pdb);b);
            if b = tt then pdb_1 := p-error-db else
              var sent_5 = p-ent-cc_findings(pdb) -_cc_findings ent_5 in
              pdb_1 := p-mk-db(p-ent-patient(pdb),
                               p-ent-doctor(pdb),
                               p-ent-cc_or(pdb),
                               p-ent-cc_data(pdb),
                               sent_5,
                               p-part_of(pdb),
                               p-orders(pdb),
                               p-determine(pdb),
                               p-make(pdb),
                               p-examination(pdb),
                               p-finding(pdb))
         end
    end
end
```

```
get-patient#(key_1, pdb; var ent_1)
begin
  if pdb = p-error-db then ent_1 := error-patient else
    ent_1 := sel-patient(key_1, p-ent-patient(pdb))
end
```

```
get-doctor#(key_2, pdb; var ent_2)
begin
  if pdb = p-error-db then ent_2 := error-doctor else
    ent_2 := sel-doctor(key_2, p-ent-doctor(pdb))
end
```

```
get-cc_or#(key_3, pdb; var ent_3)
begin
  if pdb = p-error-db then ent_3 := error-cc_or else
    ent_3 := sel-cc_or(key_3, p-ent-cc_or(pdb))
end
```

```
get-cc_data#(key_4, pdb; var ent_4)
begin
  if pdb = p-error-db then ent_4 := error-cc_data else
    ent_4 := sel-cc_data(key_4, p-ent-cc_data(pdb))
end
```

```
get-cc_findings#(key5, pdb; var ent5)
begin
   if pdb = p-error-db then ent5 := error-cc_findings else
     ent5 := sel-cc_findings(key5, p-ent-cc_findings(pdb))
end
```

```
update-patient#(key1, ent1, pdb; var pdb1)
begin
   if key-patient(ent1) ≠ key1 ∨ ent1 = error-patient then pdb1 := p-error-db else
     var ent1-1 = error-patient in
     begin
        get-patient#(key1, pdb;ent1-1);
        if ent1-1 = error-patient then pdb1 := p-error-db else
          var sent1 = p-ent-patient(pdb) -patient ent1-1 +patient ent1 in
          pdb1 := p-mk-db(sent1,
                          p-ent-doctor(pdb),
                          p-ent-cc_or(pdb),
                          p-ent-cc_data(pdb),
                          p-ent-cc_findings(pdb),
                          p-part_of(pdb),
                          p-orders(pdb),
                          p-determine(pdb),
                          p-make(pdb),
                          p-examination(pdb),
                          p-finding(pdb))
     end
end
```

```
update-doctor#(key2, ent2, pdb; var pdb1)
begin
   if key-doctor(ent2) ≠ key2 ∨ ent2 = error-doctor then pdb1 := p-error-db else
     var ent2-1 = error-doctor in
     begin
        get-doctor#(key2, pdb;ent2-1);
        if ent2-1 = error-doctor then pdb1 := p-error-db else
          var sent2 = p-ent-doctor(pdb) -doctor ent2-1 +doctor ent2 in
          pdb1 := p-mk-db(p-ent-patient(pdb),
                          sent2,
                          p-ent-cc_or(pdb),
                          p-ent-cc_data(pdb),
                          p-ent-cc_findings(pdb),
                          p-part_of(pdb),
                          p-orders(pdb),
                          p-determine(pdb),
                          p-make(pdb),
                          p-examination(pdb),
                          p-finding(pdb))
     end
end
```

```
update-cc_or#(key3, ent3, pdb; var pdb1)
begin
  if key-cc_or(ent3) ≠ key3 ∨ ent3 = error-cc_or then pdb1 := p-error-db else
    var ent3-1 = error-cc_or in
    begin
      get-cc_or#(key3, pdb;ent3-1);
      if ent3-1 = error-cc_or then pdb1 := p-error-db else
        var sent3 = p-ent-cc_or(pdb) -cc_or ent3-1 +cc_or ent3 in
        pdb1 := p-mk-db(p-ent-patient(pdb),
                        p-ent-doctor(pdb),
                        sent3,
                        p-ent-cc_data(pdb),
                        p-ent-cc_findings(pdb),
                        p-part_of(pdb),
                        p-orders(pdb),
                        p-determine(pdb),
                        p-make(pdb),
                        p-examination(pdb),
                        p-finding(pdb))
    end
end
```

```
update-cc_data#(key4, ent4, pdb; var pdb1)
begin
  if key-cc_data(ent4) ≠ key4 ∨ ent4 = error-cc_data then pdb1 := p-error-db else
    var ent4-1 = error-cc_data in
    begin
      get-cc_data#(key4, pdb;ent4-1);
      if ent4-1 = error-cc_data then pdb1 := p-error-db else
        var sent4 = p-ent-cc_data(pdb) -cc_data ent4-1 +cc_data ent4 in
        pdb1 := p-mk-db(p-ent-patient(pdb),
                        p-ent-doctor(pdb),
                        p-ent-cc_or(pdb),
                        sent4,
                        p-ent-cc_findings(pdb),
                        p-part_of(pdb),
                        p-orders(pdb),
                        p-determine(pdb),
                        p-make(pdb),
                        p-examination(pdb),
                        p-finding(pdb))
    end
end
```

```
update-cc_findings#(key5, ent5, pdb; var pdb1)
begin
  if key-cc_findings(ent5) ≠ key5 ∨ ent5 = error-cc_findingsthen pdb1 := p-error-db else
    var ent5-1 = error-cc_findings in
    begin
      get-cc_findings#(key5, pdb;ent5-1);
      if ent5-1 = error-cc_findings then pdb1 := p-error-db else
        var sent5 = p-ent-cc_findings(pdb) -cc_findings ent5-1 +cc_findings ent5 in
        pdb1 := p-mk-db(p-ent-patient(pdb),
```

```
                              p-ent-doctor(pdb),
                              p-ent-cc_or(pdb),
                              p-ent-cc_data(pdb),
                              sent5,
                              p-part_of(pdb),
                              p-orders(pdb),
                              p-determine(pdb),
                              p-make(pdb),
                              p-examination(pdb),
                              p-finding(pdb))
   end
end


est-part_of#(pdb, ent1-1, ent3-2; var pdb1)
begin
  if pdb = p-error-db ∨ ent1-1 = error-patient ∨ ent3-2 = error-cc_or then
    pdb1 := p-error-db
  else
    if sel-patient(key-patient(ent1-1), p-ent-patient(pdb)) = error-patient
      ∨ sel-cc_or(key-cc_or(ent3-2),
                              p-ent-cc_or(pdb)) = error-cc_or then
      pdb1 := p-error-db
    else
      var rel1 = p-part_of(pdb)
              +part_of mk-part_of(key-patient(ent1-1),
                                key-cc_or(ent3-2)) in
      pdb1 := p-mk-db(p-ent-patient(pdb),
                      p-ent-doctor(pdb),
                      p-ent-cc_or(pdb),
                      p-ent-cc_data(pdb),
                      p-ent-cc_findings(pdb),
                      rel1,
                      p-orders(pdb),
                      p-determine(pdb),
                      p-make(pdb),
                      p-examination(pdb),
                      p-finding(pdb))
end


est-orders#(pdb, ent2-1, ent3-2; var pdb1)
begin
  if pdb = p-error-db ∨ ent2-1 = error-doctor ∨ ent3-2 = error-cc_or then
    pdb1 := p-error-db
  else
    if sel-doctor(key-doctor(ent2-1), p-ent-doctor(pdb)) = error-doctor
      ∨ sel-cc_or(key-cc_or(ent3-2),
                  p-ent-cc_or(pdb)) = error-cc_or then
      pdb1 := p-error-db
    else
      var rel2 = p-orders(pdb)
              +orders mk-orders(key-doctor(ent2-1),
                                        key-cc_or(ent3-2)) in
      pdb1 := p-mk-db(p-ent-patient(pdb),
```

```
                    p-ent-doctor(pdb),
                    p-ent-cc_or(pdb),
                    p-ent-cc_data(pdb),
                    p-ent-cc_findings(pdb),
                    p-part_of(pdb),
                    rel₂,
                    p-determine(pdb),
                    p-make(pdb),
                    p-examination(pdb),
                    p-finding(pdb))
end
```

```
est-determine#(pdb, ent2-₁, ent4-₂; var pdb₁)
begin
  if pdb = p-error-db ∨ ent2-₁ = error-doctor ∨ ent4-₂ = error-cc_data then
    pdb₁ := p-error-db
  else
    if sel-doctor(key-doctor(ent2-₁), p-ent-doctor(pdb)) = error-doctor
      ∨ sel-cc_data(key-cc_data(ent4-₂),
                   p-ent-cc_data(pdb)) = error-cc_data then
      pdb₁ := p-error-db
    else
      var rel₃ = p-determine(pdb)
               +_determine mk-determine(key-doctor(ent2-₁),
                                        key-cc_data(ent4-₂)) in
      pdb₁ := p-mk-db(p-ent-patient(pdb),
                    p-ent-doctor(pdb),
                    p-ent-cc_or(pdb),
                    p-ent-cc_data(pdb),
                    p-ent-cc_findings(pdb),
                    p-part_of(pdb),
                    p-orders(pdb),
                    rel₃,
                    p-make(pdb),
                    p-examination(pdb),
                    p-finding(pdb))
end
```

```
est-make#(pdb, ent2-₁, ent5-₂; var pdb₁)
begin
  if pdb = p-error-db ∨ ent2-₁ = error-doctor ∨ ent5-₂ = error-cc_findings then
    pdb₁ := p-error-db
  else
    if sel-doctor(key-doctor(ent2-₁), p-ent-doctor(pdb)) = error-doctor
      ∨ sel-cc_findings(key-cc_findings(ent5-₂),
                       p-ent-cc_findings(pdb)) = error-cc_findings then
      pdb₁ := p-error-db
    else
      var rel₄ = p-make(pdb)
               +_make mk-make(key-doctor(ent2-₁),
                              key-cc_findings(ent5-₂)) in
      pdb₁ := p-mk-db(p-ent-patient(pdb),
                    p-ent-doctor(pdb),
```

```
                              p-ent-cc_or(pdb),
                              p-ent-cc_data(pdb),
                              p-ent-cc_findings(pdb),
                              p-part_of(pdb),
                              p-orders(pdb),
                              p-determine(pdb),
                              rel_4,
                              p-examination(pdb),
                              p-finding(pdb))
end



est-examination#(pdb, ent3-_1, ent4-_2; var pdb_1)
begin
  if pdb = p-error-db ∨ ent3-_1 = error-cc_or ∨ ent4-_2 = error-cc_data then
    pdb_1 := p-error-db
  else
    if sel-cc_or(key-cc_or(ent3-_1), p-ent-cc_or(pdb)) = error-cc_or
      ∨ sel-cc_data(key-cc_data(ent4-_2),
                                    p-ent-cc_data(pdb)) = error-cc_data then
      pdb_1 := p-error-db
    else
      var rel_5 = p-examination(pdb)
                 +_examination mk-examination(key-cc_or(ent3-_1),
                                              key-cc_data(ent4-_2)) in
      pdb_1 := p-mk-db(p-ent-patient(pdb),
                              p-ent-doctor(pdb),
                              p-ent-cc_or(pdb),
                              p-ent-cc_data(pdb),
                              p-ent-cc_findings(pdb),
                              p-part_of(pdb),
                              p-orders(pdb),
                              p-determine(pdb),
                              p-make(pdb),
                              rel_5,
                              p-finding(pdb))
end



est-finding#(pdb, ent4-_1, ent5-_2; var pdb_1)
begin
  if pdb = p-error-db ∨ ent4-_1 = error-cc_data ∨ ent5-_2 = error-cc_findings then
    pdb_1 := p-error-db
  else
    if sel-cc_data(key-cc_data(ent4-_1), p-ent-cc_data(pdb)) = error-cc_data
      ∨ sel-cc_findings(key-cc_findings(ent5-_2),
                             p-ent-cc_findings(pdb)) = error-cc_findings then
      pdb_1 := p-error-db
    else
      var rel_6 = p-finding(pdb)
                 +_finding mk-finding(key-cc_data(ent4-_1),
                                      key-cc_findings(ent5-_2)) in
      pdb_1 := p-mk-db(p-ent-patient(pdb),
                              p-ent-doctor(pdb),
                              p-ent-cc_or(pdb),
```

```
                        p-ent-cc_data(pdb),
                        p-ent-cc_findings(pdb),
                        p-part_of(pdb),
                        p-orders(pdb),
                        p-determine(pdb),
                        p-make(pdb),
                        p-examination(pdb),
                        rel_6)
end
```

```
rel-part_of#(pdb, ent1-_1, ent3-_2; var pdb_1)
begin
  if pdb = p-error-db ∨ ent1-_1 = error-patient ∨ ent3-_2 = error-cc_or then
    pdb_1 := p-error-db
  else
    var rel_1 = p-part_of(pdb)
                -_part_of mk-part_of(key-patient(ent1-_1),
                                     key-cc_or(ent3-_2)) in
    pdb_1 := p-mk-db(p-ent-patient(pdb),
                     p-ent-doctor(pdb),
                     p-ent-cc_or(pdb),
                     p-ent-cc_data(pdb),
                     p-ent-cc_findings(pdb),
                     rel_1,
                     p-orders(pdb),
                     p-determine(pdb),
                     p-make(pdb),
                     p-examination(pdb),
                     p-finding(pdb))
end
```

```
rel-orders#(pdb, ent2-_1, ent3-_2; var pdb_1)
begin
  if pdb = p-error-db ∨ ent2-_1 = error-doctor ∨ ent3-_2 = error-cc_or then
    pdb_1 := p-error-db
  else
    var rel_2 = p-orders(pdb)
                -_orders mk-orders(key-doctor(ent2-_1),
                                   key-cc_or(ent3-_2)) in
    pdb_1 := p-mk-db(p-ent-patient(pdb),
                     p-ent-doctor(pdb),
                     p-ent-cc_or(pdb),
                     p-ent-cc_data(pdb),
                     p-ent-cc_findings(pdb),
                     p-part_of(pdb),
                     rel_2,
                     p-determine(pdb),
                     p-make(pdb),
                     p-examination(pdb),
                     p-finding(pdb))
end
```

```
rel-determine#(pdb, ent2-_1, ent4-_2; var pdb_1)
begin
  if pdb = p-error-db ∨ ent2-_1 = error-doctor ∨ ent4-_2 = error-cc_data then
    pdb_1 := p-error-db
  else
    var rel_3 = p-determine(pdb)
                -_determine mk-determine(key-doctor(ent2-_1),
                                         key-cc_data(ent4-_2)) in
    pdb_1 := p-mk-db(p-ent-patient(pdb),
                     p-ent-doctor(pdb),
                     p-ent-cc_or(pdb),
                     p-ent-cc_data(pdb),
                     p-ent-cc_findings(pdb),
                     p-part_of(pdb),
                     p-orders(pdb),
                     rel_3,
                     p-make(pdb),
                     p-examination(pdb),
                     p-finding(pdb))
end
```

```
rel-make#(pdb, ent2-_1, ent5-_2; var pdb_1)
begin
  if pdb = p-error-db ∨ ent2-_1 = error-doctor ∨ ent5-_2 = error-cc_findings then
    pdb_1 := p-error-db
  else
    var rel_4 = p-make(pdb)
                -_make mk-make(key-doctor(ent2-_1),
                               key-cc_findings(ent5-_2)) in
    pdb_1 := p-mk-db(p-ent-patient(pdb),
                     p-ent-doctor(pdb),
                     p-ent-cc_or(pdb),
                     p-ent-cc_data(pdb),
                     p-ent-cc_findings(pdb),
                     p-part_of(pdb),
                     p-orders(pdb),
                     p-determine(pdb),
                     rel_4,
                     p-examination(pdb),
                     p-finding(pdb))
end
```

```
rel-examination#(pdb, ent3-_1, ent4-_2; var pdb_1)
begin
  if pdb = p-error-db ∨ ent3-_1 = error-cc_or ∨ ent4-_2 = error-cc_data then
    pdb_1 := p-error-db
  else
    var rel_5 = p-examination(pdb)
                -_examination mk-examination(key-cc_or(ent3-_1),
                                             key-cc_data(ent4-_2)) in
    pdb_1 := p-mk-db(p-ent-patient(pdb),
                     p-ent-doctor(pdb),
                     p-ent-cc_or(pdb),
```

```
                        p-ent-cc_data(pdb),
                        p-ent-cc_findings(pdb),
                        p-part_of(pdb),
                        p-orders(pdb),
                        p-determine(pdb),
                        p-make(pdb),
                        rel5,
                        p-finding(pdb))
end
```

```
rel-finding#(pdb, ent4-1, ent5-2; var pdb1)
begin
   if pdb = p-error-db ∨ ent4-1 = error-cc_data ∨ ent5-2 = error-cc_findings then
     pdb1 := p-error-db
   else
     var rel6 = p-finding(pdb)
                 -finding mk-finding(key-cc_data(ent4-1),
                                     key-cc_findings(ent5-2)) in
     pdb1 := p-mk-db(p-ent-patient(pdb),
                        p-ent-doctor(pdb),
                        p-ent-cc_or(pdb),
                        p-ent-cc_data(pdb),
                        p-ent-cc_findings(pdb),
                        p-part_of(pdb),
                        p-orders(pdb),
                        p-determine(pdb),
                        p-make(pdb),
                        p-examination(pdb),
                        rel6)
end
```

```
part_of#(pdb, ent1-1, ent3-2; var b)
begin
   if pdb = p-error-db ∨ ent1-1 = error-patient ∨ ent3-2 = error-cc_or then b := ff else
     if mk-part_of(key-patient(ent1-1), key-cc_or(ent3-2))
        in-part_of p-part_of(pdb) then
       b := tt
     else
       b := ff
end
```

```
orders#(pdb, ent2-1, ent3-2; var b)
begin
   if pdb = p-error-db ∨ ent2-1 = error-doctor ∨ ent3-2 = error-cc_or then b := ff else
     if mk-orders(key-doctor(ent2-1), key-cc_or(ent3-2))
        in-orders p-orders(pdb) then
       b := tt
     else
       b := ff
end
```

```
determine#(pdb, ent2-1, ent4-2; var b)
begin
   if pdb = p-error-db ∨ ent2-1 = error-doctor ∨ ent4-2 = error-cc_data then b := ff else
     if mk-determine(key-doctor(ent2-1), key-cc_data(ent4-2))
        in-determine p-determine(pdb) then
       b := tt
     else
       b := ff
end
```

```
make#(pdb, ent2-1, ent5-2; var b)
begin
   if pdb = p-error-db ∨ ent2-1 = error-doctor ∨ ent5-2 = error-cc_findings then b := ff else
     if mk-make(key-doctor(ent2-1), key-cc_findings(ent5-2))
        in-make p-make(pdb) then
       b := tt
     else
       b := ff
end
```

```
examination#(pdb, ent3-1, ent4-2; var b)
begin
   if pdb = p-error-db ∨ ent3-1 = error-cc_or ∨ ent4-2 = error-cc_data then b := ff else
     if mk-examination(key-cc_or(ent3-1), key-cc_data(ent4-2))
        in-examination p-examination(pdb) then
       b := tt
     else
       b := ff
end
```

```
finding#(pdb, ent4-1, ent5-2; var b)
begin
   if pdb = p-error-db ∨ ent4-1 = error-cc_data ∨ ent5-2 = error-cc_findings then b := ff else
     if mk-finding(key-cc_data(ent4-1), key-cc_findings(ent5-2))
        in-finding p-finding(pdb) then
       b := tt
     else
       b := ff
end
```

```
legal-part_of#(rel1, sent1, sent3; var b)
begin
   if rel1 = emptyrel-part_of then b := tt else
     var pair1 = min-part_of(rel1) in
     if sel-patient(fst-part_of(pair1), sent1) = error-patient
        ∨ sel-cc_or(snd-part_of(pair1), sent3) = error-cc_or then
       b := ff
     else
       legal-part_of#(rest-part_of(rel1), sent1, sent3;b)
end
```

```
legal-orders#(rel_2, sent_2, sent_3; var b)
begin
   if rel_2 = emptyrel-orders then b := tt else
     var pair_2 = min-orders(rel_2) in
     if sel-doctor(fst-orders(pair_2), sent_2) = error-doctor
        ∨ sel-cc_or(snd-orders(pair_2), sent_3) = error-cc_or then
       b := ff
     else
        legal-orders#(rest-orders(rel_2), sent_2, sent_3;b)
end
```

```
legal-determine#(rel_3, sent_2, sent_4; var b)
begin
   if rel_3 = emptyrel-determine then b := tt else
     var pair_3 = min-determine(rel_3) in
     if sel-doctor(fst-determine(pair_3), sent_2) = error-doctor
        ∨ sel-cc_data(snd-determine(pair_3), sent_4) = error-cc_data then
       b := ff
     else
        legal-determine#(rest-determine(rel_3), sent_2, sent_4;b)
end
```

```
legal-make#(rel_4, sent_2, sent_5; var b)
begin
   if rel_4 = emptyrel-make then b := tt else
     var pair_4 = min-make(rel_4) in
     if sel-doctor(fst-make(pair_4), sent_2) = error-doctor
        ∨ sel-cc_findings(snd-make(pair_4), sent_5) = error-cc_findings then
       b := ff
     else
        legal-make#(rest-make(rel_4), sent_2, sent_5;b)
end
```

```
legal-examination#(rel_5, sent_3, sent_4; var b)
begin
   if rel_5 = emptyrel-examination then b := tt else
     var pair_5 = min-examination(rel_5) in
     if sel-cc_or(fst-examination(pair_5), sent_3) = error-cc_or
        ∨ sel-cc_data(snd-examination(pair_5), sent_4) = error-cc_data then
       b := ff
     else
        legal-examination#(rest-examination(rel_5), sent_3, sent_4;b)
end
```

```
legal-finding#(rel_6, sent_4, sent_5; var b)
begin
   if rel_6 = emptyrel-finding then b := tt else
     var pair_6 = min-finding(rel_6) in
     if sel-cc_data(fst-finding(pair_6), sent_4) = error-cc_data
        ∨ sel-cc_findings(snd-finding(pair_6), sent_5) = error-cc_findings then
```

```
      b := ff
    else
      legal-finding#(rest-finding(rel6), sent4, sent5;b)
end
```

```
in-fst-part_of#(k1-1, rel1; var b)
begin
  if rel1 = emptyrel-part_of then b := ff else
    if k1-1 = fst-part_of(min-part_of(rel1)) then b := tt else
      in-fst-part_of#(k1-1, rest-part_of(rel1);b)
end
```

```
in-fst-orders#(k2-1, rel2; var b)
begin
  if rel2 = emptyrel-orders then b := ff else
    if k2-1 = fst-orders(min-orders(rel2)) then b := tt else
      in-fst-orders#(k2-1, rest-orders(rel2);b)
end
```

```
in-fst-determine#(k3-1, rel3; var b)
begin
  if rel3 = emptyrel-determine then b := ff else
    if k3-1 = fst-determine(min-determine(rel3)) then b := tt else
      in-fst-determine#(k3-1, rest-determine(rel3);b)
end
```

```
in-fst-make#(k4-1, rel4; var b)
begin
  if rel4 = emptyrel-make then b := ff else
    if k4-1 = fst-make(min-make(rel4)) then b := tt else
      in-fst-make#(k4-1, rest-make(rel4);b)
end
```

```
in-fst-examination#(k5-1, rel5; var b)
begin
  if rel5 = emptyrel-examination then b := ff else
    if k5-1 = fst-examination(min-examination(rel5)) then
      b := tt
    else
      in-fst-examination#(k5-1, rest-examination(rel5);b)
end
```

```
in-fst-finding#(k6-1, rel6; var b)
begin
  if rel6 = emptyrel-finding then b := ff else
    if k6-1 = fst-finding(min-finding(rel6)) then b := tt else
      in-fst-finding#(k6-1, rest-finding(rel6);b)
end
```

```
in-snd-part_of#(k1-_2, rel_1; var b)
begin
  if rel_1 = emptyrel-part_of then b := ff else
    if k1-_2 = snd-part_of(min-part_of(rel_1)) then b := tt else
      in-snd-part_of#(k1-_2, rest-part_of(rel_1);b)
end
```

```
in-snd-orders#(k2-_2, rel_2; var b)
begin
  if rel_2 = emptyrel-orders then b := ff else
    if k2-_2 = snd-orders(min-orders(rel_2)) then b := tt else
      in-snd-orders#(k2-_2, rest-orders(rel_2);b)
end
```

```
in-snd-determine#(k3-_2, rel_3; var b)
begin
  if rel_3 = emptyrel-determine then b := ff else
    if k3-_2 = snd-determine(min-determine(rel_3)) then b := tt else
      in-snd-determine#(k3-_2, rest-determine(rel_3);b)
end
```

```
in-snd-make#(k4-_2, rel_4; var b)
begin
  if rel_4 = emptyrel-make then b := ff else
    if k4-_2 = snd-make(min-make(rel_4)) then b := tt else
      in-snd-make#(k4-_2, rest-make(rel_4);b)
end
```

```
in-snd-examination#(k5-_2, rel_5; var b)
begin
  if rel_5 = emptyrel-examination then b := ff else
    if k5-_2 = snd-examination(min-examination(rel_5)) then
      b := tt
    else
      in-snd-examination#(k5-_2, rest-examination(rel_5);b)
end
```

```
in-snd-finding#(k6-_2, rel_6; var b)
begin
  if rel_6 = emptyrel-finding then b := ff else
    if k6-_2 = snd-finding(min-finding(rel_6)) then b := tt else
      in-snd-finding#(k6-_2, rest-finding(rel_6);b)
end
```

```
rs#(pdb)
begin
  if pdb = p-error-db then skip else
    var pdb_1 = p-error-db in
    begin
      mk-db#(p-ent-patient(pdb),
             p-ent-doctor(pdb),
             p-ent-cc_or(pdb),
             p-ent-cc_data(pdb),
             p-ent-cc_findings(pdb),
             p-part_of(pdb),
             p-orders(pdb),
             p-determine(pdb),
             p-make(pdb),
             p-examination(pdb),
             p-finding(pdb);
             pdb_1);
      if pdb_1 = p-error-db then abort
    end
end
```

Bibliography

[Aut93] S. Autexier. Hdms-A und Obscure in Korso — Die funktionale Essenz von Hdms-A aus Sicht der algorithmischen Spezifikationsmethode. Teil 2: Schablonen zur Übersetzung eines E/R-Schemas in eine Obscure Spezifikation. Technical Report A/05/93, Universität des Saarlandes, Saarbrücken, December 1993.

[Ben93] Ch. Benzmüller. Hdms-A und Obscure in Korso — Die funktionale Essenz von Hdms-A aus Sicht der algorithmischen Spezifikationsmethode. Teil 3: Die Spezifikation der atomaren Funktionen. Technical Report A/06/93, Universität des Saarlandes, Saarbrücken, December 1993.

[BFG+93] M. Broy, C. Facchi, R. Grosu, R. Hettler, H. Hußmann, D. Nazareth, F. Regensburger, and K. Stølen. The Requirement and Design Specification Language Spectrum — An Informal Introduction, Version 1.0. Technical report, Technische Universität München, 1993.

[BS93] J. Bohn and M. Schulte. Entwicklung von Spezifikationen für das Hdms-A Basismodell aus einer gegebenen Anfangsspezifikation mittels Transformation. Interim Report of the Diploma Thesis, 55 pages, Universität Oldenburg, December 1993.

[CFLW92] F. Cornelius, J. Faulhaber, M. Löwe, and R. Wessäly. Ein Fallbeispiel für Korso: Das heterogene verteilte Managementsystem Hdms der Projektgruppe Medizin Informatik (PMI) am Deutschen Herzzentrum Berlin und an der TU Berlin — Ein Vorschlag. Report 92–45, TU Berlin, FB Informatik, December 1992.

[CHL94a] F. Cornelius, H. Hußmann, and M. Löwe. The Korso Case Study for Software Engineering with Formal Methods: A Medical Information System. In M. Broy and S. Jähnichen, editors, *Korso, Correct Software by Formal Methods*, Lecture Notes in Computer Science. Springer, 1994. to appear, also published in an extended version as technical report no. 94-5, Technische Universität Berlin, February 1994.

[CHL94b] F. Cornelius, H. Hußmann, and M. Löwe. The Korso Case Study for Software Engineering with Formal Methods: A Medical Information System. Technical Report 94-5, Technische Universität Berlin, February 1994. to appear.

[CKL93] F. Cornelius, M. Klar, and M. Löwe. Ein Fallbeispiel für Korso: Ist-Analyse Hdms-A. Technical Report 93-28, Technische Universität Berlin, 1993.

[Con93a] S. Conrad. Einbindung eines bestehenden Datenbanksystems in einen formalen Software-Entwicklungsprozeß — ein Beitrag zur Hdms-A-Fallstudie. In H.-D. Ehrich, editor, *Beiträge zu Korso- und Troll light-Fallstudien*, pages 1–14. Technische Universität Braunschweig, Informatik-Bericht 93–11, 1993.

[Con93b] S. Conrad. Spezifikation eines vereinfachten Datenbanksystems — ein Beitrag zur Hdms-A-Fallstudie. In H.-D. Ehrich, editor, *Beiträge zu Korso- und Troll light-Fallstudien*, pages 15–26. Technische Universität Braunschweig, Informatik-Bericht 93–11, 1993.

[Dam93] W. Damm. Korso-Applikationen — Hdms-A Teilprojekt Universität Oldenburg. Short description of ongoing work, February 1993.

[GH93] M. Grosse and H. Hufschmidt. SOLL-Spezifikation aus Sicht der Sicherheit. Technical Report A/07/93, Universität des Saarlandes, Saarbrücken, December 1993.

[Hec93] A. Heckler. HDMS-A und OBSCURE in KORSO — Die funktionale Essenz von HDMS-A aus Sicht der algorithmischen Spezifikationsmethode. Teil 1: Einführung und Anmerkungen. Technical Report A/04/93, Universität des Saarlandes, Saarbrücken, December 1993.

[Het93] R. Hettler. Übersetzung von E/R-Modellen nach SPECTRUM. Technical Report TUM-I9333, Technische Universität München, 80290 München, Germany, 1993. (in German).

[HHM+93] R. Hettler, H. Hußmann, S. Merz, F. Nickl, and O. Slotosch. Die funktionale Essenz von HDMS-A. Technical Report TUM-I9335, Technische Universität München, 1993.

[HRS87] M. Heisel, W. Reif, and W. Stephan. Program Verification by Symbolic Execution and Induction. In K. Morik, editor, *Proc. 11th German Workshop on Artifical Intelligence*, number 152 in Informatik Fachberichte. Springer, 1987.

[HRS89] M. Heisel, W. Reif, and W. Stephan. A Dynamic Logic for Program Verification. In A. Meyer and M. Taitslin, editors, *Logical Foundations of Computer Science*, volume 363 of *Lecture Notes in Computer Science*, pages 134–145. Logic at Botik, Pereslavl-Zalessky, Russia, Springer, 1989.

[HRS90] M. Heisel, W. Reif, and W. Stephan. Tactical Theorem Proving in Program Verification. In *10th International Conference on Automated Deduction*, volume 449 of *Lecture Notes in Artificial Intelligence.* Kaiserslauter, Germany, Springer, July 1990.

[HS93] A. Heckler and M. Strecker. Modifizierungsrahmen, Zwei-Ebenen-Konzept und Eintopf-Konzept — Erster Bericht der Rahmen-Gruppe — vormals "'Erweiterbarkeitsgruppe"'. Technical Report at the Universities of Ulm and Saarbrücken, to appear, March 1993.

[Huß93] H. Hußmann. Zur formalen Beschreibung der funktionalen Anforderungen an ein Informationssystem. Technical Report TUM-I9332, Technische Universität München, 80290 München, Germany, 1993.

[MZ94] M. Mehlich and W. Zhang. Specifying Interactive Components for Configurating Graphical User Interfaces. Technical Report at the Ludwig-Maximilian Universität Munich, to appear, 1994.

[Nic93] F. Nickl. Ablaufspezifikation durch Datenflußmodellierung und stromverarbeitende Funktionen. Technical Report TUM-I9334, Technische Universität München, 1993.

[Rei92] Wolfgang Reif. Verification of Large Software Systems. In Shyamasundar, editor, *proc. Foundations of Software Technology and Theoretical Computer Science*, Lecture Notes in Computer Science. New Dehli, India, Springer, 1992.

[Ren94] K. Renzel. Formale Beschreibung von Sicherheitsaspekten für das Fallbeispiel HDMS-A. Technical Report 9402, Ludwig-Maximilians Universität Munich, January 1994.

[Sch89] Gerhard Schellhorn. Examples for the Verification of Modules in Dynamic Logic. Technical report, Institut für Logik, Komplexität und Deduktionssysteme, Fakultät für Informatik, Universität Karlsruhe, Germany, 1989. (in German).

[Shi94] H. Shi. Benutzerschnittstelle und -Interaktion für die HK-Untersuchung. Technical Report at the Universität Bremen, to appear, February 1994.

[Ste93] Kurt Stenzel. A Verified Access Control Model. Internal Note 26/93, Institut für Logik, Komplexität und Deduktionssysteme, Fakultät für Informatik,Universität Karlsruhe, 76128 Karlsruhe, Germany, December 1993.

[tB92] Johan ter Bekke. *Semantic Data Modeling.* PRENTICE HALL, 1992.