Proof of the Basic Theorem on Concept Lattices in Isabelle/HOL

Barış Sertkaya[1,2] and Halit Oğuztüzün[2]

[1] Institute of Theoretical Computer Science, Dresden University of Technology
Dresden, Germany
[2] Department of Computer Engineering, Middle East Technical University
Ankara, Turkey

Abstract. This paper presents a machine-checked proof of the Basic Theorem on Concept Lattices, which appears in the book "Formal Concept Analysis" by Ganter and Wille, in the Isabelle/HOL Proof Assistant. As a by-product, the underlying lattice theory by Kammueller has been extended.

1 Introduction

Formal Concept Analysis (FCA) [1] is an emerging field of applied mathematics based on a lattice-theoretic formalization of the notions of concept and conceptual hierarchy. It thereby facilitates mathematical reasoning for conceptual data analysis and knowledge processing. In FCA, a concept is constituted by two parts: its *extent*, which consists of all the objects belonging to the concept, and its *intent*, which contains the attributes common to all objects of the concept. This formalization allows the user to form all concepts of a context and introduce a subsumption hierarchy between the concepts, resulting in a complete lattice called the *concept lattice* of the context. Concept lattice is used to query the knowledge and to derive implicit information from the knowledge.

Isabelle [2, 3], on the other hand, is a generic interactive theory development environment for implementing logical formalisms. It has been instantiated to support reasoning in several object-logics. Specialization of Isabelle for Higher Order Logic is called Isabelle/HOL.

The long term goal of this effort is to formalize the theory of FCA in Isabelle/HOL. This will provide a mechanized theory for researchers to prove their own theorems with utmost precision and to verify the knowledge representation systems based on FCA. Another potential utility of formalization is extracting programs from constructive proofs. See, for example, [17, 18].

The specific accomplishment of this work is a machine-checked version of the proof of the Basic Theorem of Concept Lattices, which appears in the book "Formal Concept Analysis" by Ganter and Wille [1]. As a by-product, the underlying lattice theory developed by Kammueller [9] has been extended.

In an effort along the same direction, Schwarzweller presents a formalization of concept lattices in Mizar Proof Assistant [12, 13]. Some applications of FCA to Knowledge Engineering and Software Engineering have been reported [14, 15, 16].

2 Isabelle Proof Assistant

Isabelle is a *generic* interactive theorem prover, designed for reasoning in a variety of formal theories. It is generic in the sense that it provides proof procedures for Constructive Type Theory, various first-order logics, some systems of Modal Logics, Zermelo-Fraenkel Set Theory, and Higher-Order Logic, which are called *object-logics*. Object-logics are formalized within Isabelle's *meta-logic*, which is intuitionistic higher-order logic with implication, universal quantifiers, and equality. The specialization of Isabelle for Higher Order Logic is called Isabelle/HOL [5]. It is a widely used object-logic for proof-checking tasks.

2.1 Isabelle Theories

Working with Isabelle/HOL means creating theories. Roughly speaking, a *theory* is a named collection of types, functions, theorems and their proofs. The general format of a theory `T` is

```
theory T = B1 + ... + Bn:
  declarations, definitions, and proofs
end
```

where `B1, ... , Bn` are the names of existing (parent) theories that `T` is based on and `declarations, definitions and proofs` represent the newly introduced concepts (types, functions etc.) and proofs of theorems. Everything defined in the parent theories (and their parents recursively) is visible.

2.2 Theorem Proving with Isabelle

Proof trees are derived rules, and are built by joining rules together. This comprises both forwards and backwards proof. A backwards proof works by matching a goal with the conclusion of a rule; the premises become the subgoals. A forwards proof works by matching theorems to the premises of rules, making a new theorem.

A typical proof starts with first stating the goal using the `Goal` command, proceeds with applying tactics aiming to solve this goal using the `by` command, and ends with the `qed` command which names and stores the proved theorem. Tactics may lead to zero or more subgoals. The proof process continues until no subgoals are left.

Isabelle/HOL has a huge number of predefined tactics. Some of the most widely used groups of tactics are resolution, rewrite, induction, assumption, tableau, automatic and simplification tactics. Apart from them, the user can define her/his own tactics. A complete list of tactics can be found in [4].

2.3 Lattice Theory in Isabelle/HOL

Our formalization is based on the theory `Tarski`, which was developed by Florian Kammueller to prove Tarski's Fixpoint Theorem. At the time this work started, the theory was available in old style proof script. It contains a minimal version of lattice theory providing partial orders, complete lattices, least upper bound, greatest lower bound and fixed points of complete lattices. The type of a partially ordered set is defined by the record type 'a potype as:

```
record 'a potype =
  pset  :: "'a set"
  order :: "('a * 'a) set"
```

The field pset is the set of elements of the partial order and the field order is the set of pairs of elements with the meaning that the first element of the pair is less than or equal to the second one. Using syntactic translations, the field pset of a partial order V is accessed as V.<A> and the field order is accessed as V.<r>. The theory provides the least upper bound and the greatest lower bound operations on a partially ordered set with the lub and glb functions respectively. Apart from these, it provides the predicates islub to check if a given element is the least upper bound of a partial order, and the predicate isglb to check if a given element is the greatest lower bound of a partial order. Using these definitions and some auxiliary definitions, the theory introduces complete lattices. In addition to these definitions, it also provides the proofs of the uniqueness of the *lub* and the *glb*, the proof that *lub* and *glb* are elements of the lattice, properties about duality, and finally the Tarski's lemma on fixpoints.

We extended the theory with the formal definitions of supremum and infimum preserving maps on complete lattices, order preserving maps, order embeddings, supremum/infimum-dense sets, supremum/infimum-irreducible elements, complete lattice homomorphism and complete lattice isomorphism. (For the Isabelle symbols appearing in the following definitions, please refer to [5], [4] or table 1 on page 9.) Since we are dealing with complete lattices, we defined supremum preserving maps on complete lattices as:

```
supremum_preserving :: "['a => 'b , 'a potype , 'b potype] => bool"
"supremum_preserving f V1 V2 == (V1 : CompleteLattice) &
  (V2 : CompleteLattice) & (f ` (V1.<A>) <= V2.<A>) &
  (! X <= V1.<A> . ! x : V1.<A> . (islub X V1 x) -->
  (islub (f ` X) V2 (f x)))"
```

Infimum preserving map infimum_preserving is defined in a similar way. Order preserving maps and order embeddings are defined as:

```
order_preserving :: "['a => 'b , 'a potype , 'b potype] => bool"
"order_preserving f V1 V2 ==  ! x : V1.<A> . ! y : V1.<A> .
  ((x,y) : V1.<r>) --> (((f x) , (f y)) : V2.<r>)"

order_embedding :: "['a => 'b , 'a potype , 'b potype] => bool"
"order_embedding f V1 V2 ==  ! x : V1.<A> . ! y : V1.<A> .
  ((x,y) : V1.<r>) = (((f x) , (f y)) : V2.<r>)"
```

Using the functions defined above, we defined a lattice homomorphism to be a supremum, infimum and order preserving map:

```
lattice_homomorphism :: "['a => 'b , 'a potype , 'b potype] => bool"
"lattice_homomorphism f V1 V2 == (supremum_preserving f V1 V2) &
  (infimum_preserving f V1 V2) & (order_preserving f V1 V2)"
```

And a lattice isomorphism to be an injective and surjective lattice homomorphism:

```
lattice_isomorphism :: "['a => 'b , 'a potype , 'b potype] => bool"
"lattice_isomorphism f V1 V2 == (lattice_homomorphism f V1 V2) &
  (inj f) & (my_surj f (V1.<A>) (V2.<A>))"
```

Since the Isabelle primitive `surj` for surjective maps does not take types into account, we defined our own typed surjective maps `my_surj` as:

```
my_surj :: "['a => 'b,'a set,'b set] => bool"
"my_surj f V1 V2 == ! y : V2 . ? x : V1 . y = (f x)"
```

And we defined supremum-dense sets as:

```
supremum_dense :: "['a set,'a potype] => bool"
"supremum_dense X V == (X <= V.<A>) & (! v : V.<A> .
  ? A <= X . islub A V v)"
```

Infimum-dense set `infimum_dense` is defined in a similar way.

In preparation for the proof of the Basic Theorem, we proved some theorems from Lattice Theory. We proved that the supremum/infimum dense property is preserved under an isomorphism. We also proved that the supremum of a subset of a set is less than or equal to the supremum of its superset. The formal proofs can be found in [11] as stand alone lemmata with names `sup_dense_preserved`, `inf_dense_preserved` and `sup_lt_ss` respectively.

3 Formalization

In this section, we present the basic notions of Formal Concept Analysis and their formalizations in Isabelle/HOL in an interleaved manner. First we give the mathematical notions as in [1], then we give the corresponding Isabelle/HOL proof script and related commentary. Due to space limitations, we can not give the proofs in full details, the interested reader may see [1] for the mathematical notions and proofs, and [11] for the corresponding Isabelle/HOL proof script and a detailed commentary of it.

We start with basic definitions, and datatypes defined for them in Isabelle/HOL.

3.1 Definitions and Datatypes

Definition 1. A **Formal Context** $\mathbb{K} := (G, M, I)$ consists of two sets G and M and a relation I between G and M. The elements of G are called the **objects** and the elements of M are called the **attributes** of the context. The I relation between an object g and an attribute m is written as gIm or $(g,m) \in I$ and read as "the object g **has** the attribute m". The relation I is also called the **incidence relation** of the context.

Using the definition, formal context type is formalized as a record type with fields object_set, attribute_set and incidence_rel as:

```
record ('a,'b) formal_context_type =
    object_set      :: "'a set"
    attribute_set   :: "'b set"
    incidence_rel   :: "('a * 'b) set"
```

Through syntactic translations, the object set of a formal context K is accessed as K.<OS>, attribute set as K.<AS> and the incidence relation as K.<IR>.

Definition 2. For a set $A \subseteq G$ of objects, the set of attributes common to the objects in A is defined as: $A' = \{m \in M \,|\, (g,m) \in I \, for\, all\, g \in A\}$. Correspondingly, for a set $B \subseteq M$, the set of objects which have all attributes in B is defined as: $B' = \{g \in G \,|\, (g,m) \in I \, for\, all\, m \in B\}$

The polymorphic prime operator is formalized as two separate functions namely common_attributes and common_objects, in the following manner:

```
common_attributes :: "'a set => ('a,'b) formal_context_type =>
   'b set"
"common_attributes os fc == {
   m . m : fc.<AS> & (! g : os . (g,m) : fc.<IR>) & os <= fc.<OS>
}"
```

common_attributes is the formal definition of a function which takes a set of objects os of type 'a set and a formal context fc of type ('a,'b) formal_context_type and returns the set of attributes (of type 'b set) common to all objects in os.

```
common_objects :: "'b set => ('a,'b) formal_context_type =>
   'a set"
"common_objects as fc == {
   g . g : fc.<OS> & (! m : as . (g,m) : fc.<IR>) & as <= fc.<AS>
}"
```

Correspondingly, common_objects is the formal definition of a function which takes a set of attributes as of type 'b set and a formal context fc of type ('a,'b) formal_context_type and returns the set of objects (of type 'a set) which have all attributes in as.

Definition 3. A **Formal Concept** of the context $\mathbb{K} := (G,M,I)$ is a pair (A,B) with $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. A is called the **extent** and B is called the **intent** of the formal concept (A,B).

From the definition, formal concept type is formalized as a record type with fields `extent` and `intent` as:

```
record ('a,'b) formal_concept_type =
    extent          :: "'a set"
    intent          :: "'b set"
```

Similarly, through syntactic translations, the extent of a formal concept `C` is accessed as `C.<E>`, and the intent as `C.<I>`. The relation between the extent and the intent of a formal concept is checked with the boolean function `FormalConcept`. Given a tuple `C` of type `formal_concept_type` and a triple `K` of type `formal_context_type`, it checks if `C` is a formal concept of `K`. It is formalized as:

```
FormalConcept :: "('a,'b) formal_concept_type =>
   ('a,'b) formal_context_type => bool"
"FormalConcept C K == C.<E> <= K.<OS> & C.<I> <= K.<AS> &
   C.<E> = common_objects (C.<I>) K &
   common_attributes (C.<E>) K = C.<I>"
```

Proposition 4. *If T is an index set and, for every $t \in T$, $A_t \subseteq G$ is a set of objects, then*

$$\left(\bigcup_{t \in T} A_t\right)' = \bigcap_{t \in T} A'_t$$

The same holds for the sets of attributes.

The proposition is formalized as:

```
Goal "[| ! t : T . (F t) <= K.<OS> |] ==> (common_attributes (
UN t : T . (F t)) K) = (INT t : T . (common_attributes (F t) K))";
```

We are not going to give the proof here, the interested reader may see [1] and [11]. But we would like to draw attention to the following point: In the proof in [1], the case where the index set T can be empty is not worked out explicitly. But in the formalization we need to do a case analysis for T being empty or not, since the set theory does not have the convention about empty index sets. This case is handled with an axiom which states that common attributes of an empty object set is equal to the attribute set of the context. Similarly, an axiom is added which states that common objects of an empty attribute set is equal to the object set of the context.

Definition 5. If (A_1, B_1) and (A_2, B_2) are concepts of a context, (A_1, B_1) is called a **subconcept** of (A_2, B_2), provided that $A_1 \subseteq A_2$ (which is equivalent to $B_2 \subseteq B_1$). In this case, (A_2, B_2) is a **superconcept** of (A_1, B_1) and the ordering is written as $(A_1, B_1) \leq (A_2, B_2)$. The relation $\leq$ is called the **hierarchical order** (or simply **order**) of the concepts. The set of all concepts of (G, M, I) ordered in this way is denoted by $\underline{\mathfrak{B}}(G, M, I)$ and is called the **Concept Lattice** of the context (G, M, I).

The concept lattice of a context `K` is formalized with the function `ConceptLattice` which takes a context `K` and returns the concept lattice of it as a partial order type:

```
ConceptLattice :: "('a,'b) formal_context_type =>
   ((('a,'b) formal_concept_type) potype)"
"ConceptLattice K == (|
   pset = {C . (FormalConcept C K)},
   order = { (C1,C2) . FormalConcept C1 K & FormalConcept C2 K &
      C1.<E> <= C2.<E> & C2.<I> <= C1.<I> } |)"
```

3.2 The Basic Theorem on Concept Lattices

Theorem 6 (The Basic Theorem on Concept Lattices). *The concept lattice $\underline{\mathfrak{B}}(G, M, I)$ is a complete lattice in which infimum and supremum are given by:*

$$\bigwedge_{t \in T} (A_t, B_t) = \left(\bigcap_{t \in T} A_t, \left(\bigcup_{t \in T} B_t \right)'' \right)$$
$$\bigvee_{t \in T} (A_t, B_t) = \left(\left(\bigcup_{t \in T} A_t, \right)'', \bigcap_{t \in T} B_t \right)$$

A complete lattice $\mathbf{V}$ is isomorphic to $\underline{\mathfrak{B}}(G, M, I)$ if and only if there are mappings $\tilde{\gamma} : G \to V$ and $\tilde{\mu} : M \to V$ such that $\tilde{\gamma}(G)$ is supremum-dense in $\mathbf{V}$, $\tilde{\mu}(M)$ is infimum-dense in $\mathbf{V}$ and gIm is equivalent to $\tilde{\gamma}g \leq \tilde{\mu}m$ for all $g \in G$ and all $m \in M$. In particular, $\mathbf{V} \cong \underline{\mathfrak{B}}(\mathbf{V}, \mathbf{V}, \leq)$

We prove the theorem in four major parts, as four lemmas. First we prove the claims about the infimum and supremum of a concept lattice, and then both directions of the double implication about the isomorphism.

The argument about the infimum is formalized as:

```
Goal "[| S <= (ConceptLattice K).<A> |] ==>
  isglb S (ConceptLattice K) (| extent = (INT C : S . C.<E>) ,
  intent = (common_attributes (common_objects
  (UN C : S . C.<I>) K) K) |)";
```

The `isglb` is a predicate from the underlying lattice theory. It checks if the third argument is the infimum of the set given as first argument in the partially ordered set given as the second argument. We start with simplifying the goal with the definition of `isglb`, and get three subgoals. First we prove that the concept argued as the infimum is in `(ConceptLattice K)`, which means to prove that it is a formal concept of the context `K`. Then we prove that it is a lower bound by showing that it is less than or equal to all other formal concepts in `S`. As the last subgoal we prove that it is the greatest lower bound. The proof is totally 29 steps not including the number of steps of the auxiliary lemma used. It is stored as `inf_cl` for further use.

Correspondingly, the argument about the supremum is formalized as:

```
Goal "[| S <= (ConceptLattice K).<A> |] ==>
  islub S (ConceptLattice K) (| extent = (common_objects
  (common_attributes (UN C : S . C.<E>) K) K) ,
  intent = (INT C : S . C.<I>) |)";
```

Similar remarks as for the preceding argument apply here. The proof is totally 32 steps without counting the number of steps of the auxiliary lemmata used. It is named and stored as `sup_cl` for further use.

Next we prove the argument about the isomorphism. First, we prove the *only if* direction of the double implication. We formalized the statement as:

```
Goal "[| V : CompleteLattice |] ==>
 (isomorphic (ConceptLattice K) V) -->
 (? gamma mu . (supremum_dense (gamma ` (K.<OS>)) V) &
 (infimum_dense (mu ` (K.<AS>)) V) &
 (! g : K.<OS> . ! m : K.<AS> . ((g,m) :
 K.<IR>) = (((gamma g),(mu m)) : V.<r>)))";
```

For the special case $\mathbf{V} = \underline{\mathfrak{B}}(\mathbf{G}, \mathbf{M}, \mathbf{I})$, we first prove that $\widetilde{\gamma}(G)$ is supremum-dense and $\widetilde{\mu}(M)$ is infimum-dense in $\underline{\mathfrak{B}}(G, M, I)$, and gIm is equivalent to $\widetilde{\gamma}g \leq \widetilde{\mu}m$ for all $g \in G$ and all $m \in M$. The first two proofs are stored as the lemmata `gamma_sup_dense` and `mu_inf_dense` respectively. Later we prove these three properties for the general case $\mathbf{V}$ is isomorphic to $\underline{\mathfrak{B}}(G, M, I)$ using the lemmata above together with the lemmata `sup_dense_preserved` and `inf_dense_preserved`. This completes the proof of the *only if* direction of the theorem. Without counting the steps of the auxiliary lemmata used, the proof is 70 steps long. It is stored as the lemma `basic_thm_fwd`.

Next we proceed with the proof of the *if* direction of the theorem. We formalized the statement as:

```
Goal "[| V : CompleteLattice |] ==>
  ? phi psi . ? gamma mu . (supremum_dense (gamma ` (K.<OS>)) V) &
  (infimum_dense (mu ` (K.<AS>)) V) & (! g : K.<OS> .
  ! m : K.<AS> . ((g,m) : K.<IR>) = (((gamma g),(mu m)) : V.<r>)) -->
  (order_preserving phi (ConceptLattice K) V) &
  (order_preserving psi V (ConceptLattice K)) &
  (my_inv phi psi ((ConceptLattice K).<A>) (V.<A>))";
```

(For this direction, the fact that isomorphism implies the order-embedding property is implicitly used in the book. But we proved this formally and stored as the lemma `iso_imp_embd`.) We start with proving that the maps φ and ψ are order-preserving. Then we prove that φ and ψ are inverse functions. We prove this in two parts; first we show that ψ is the left-inverse of φ, then we show that it is also the right-inverse of φ. Having proved that φ and ψ are order-preserving inverse maps, we proved that φ is a lattice isomorphism. This completes the `if` direction of the proof. It is stored as the lemma `basic_theorem_bwd`. It is 286 steps without counting the steps of the auxiliary lemmata used.

4 Conclusion and Discussions

Although mathematics texts typically do not give the proofs in whole detail, they are understandable by human reader. In an informal proof, some details of the proof can be skipped relying on human intuition. But for a proof to be machine-checkable, every single step of it has to be stated formally. There should not be any gaps between proof steps, however minor they might be.

During our formalization, we noticed some of these kinds of gaps in the proofs. We have already mentioned the implicit treatment of empty index sets, empty object sets and empty attribute sets in the book. Furthermore, for connecting the proofs of the *only if* and *if* parts of the basic theorem, a lemma from lattice theory is used but it is not mentioned clearly, since it is supposedly well-known to mathematicians.

Separately, we examined the proof of basic theorem in the formal concept analysis chapter of a well-known book [10]. There, in the proof of *only if* direction of the basic theorem on page 71, we have uncovered a mistake apparently arising from misuse of overloaded symbols. It is written that the statement 'gIm if and only if $\widetilde{\gamma}(g) \leq \widetilde{\mu}(m)$ is in $\underline{\mathfrak{B}}(G, M, I)$, for all g in G and for all m in M' is proved in 3.7. But the proof in 3.7 corresponds to the proof of the third subgoal of the *only if* direction of the Basic Theorem in the special case $\mathbf{L}$ is equal to $\underline{\mathfrak{B}}(G, M, I)$. So it does not constitute a proof in the general case $\underline{\mathfrak{B}}(G, M, I)$ is isomorphic to $\mathbf{L}$. We think this part of the proof should be generalized to the isomorphism case. This is a testimony to the utility of formalization in revealing hidden gaps in published proofs.

Table 1. Notation Index

Math. Notation	Isabelle Notation	Definition
$'$ (*polymorphic*)	`common_attributes`	Common Attributes of an object set
$'$ (*polymorphic*)	`common_objects`	Common Objects of an attribute set
(G, M, I)	`K`	Context K
$\underline{\mathfrak{B}}(G, M, I)$	`(ConceptLattice K)`	Concept Lattice of the context K
$\bigwedge_{t \in T}(A_t, B_t)$	`(glb S K)`	Infimum of `S` in `K`
$\bigvee_{t \in T}(A_t, B_t)$	`(lub S K)`	Supremum of `S` in `K`
$\in$	`:`	In
$\wedge$	`&`	Conjunction
$\vee$	`|`	Disjunction
$\longrightarrow$	`-->`	Implication
$\forall t \in T. P(t)$	`! t : T . (P t)`	Universal Quantifier
$\exists t \in T. P(t)$	`? t : T . (P t)`	Existential Quantifier
$\subseteq$	`<=`	Subset or equal
$\bigcup_{t \in T} A_t$	`UN t : T . (F t)`	Indexed set union
$\bigcap_{t \in T} A_t$	`INT t : T . (F t)`	Indexed set intersection

Acknowledgments

The authors appreciate the help they received from the members of the isabelle-users mailing list, particularly Larry Paulson and Tobias Nipkow.

References

1. B. Ganter, R. Wille: Formal Concept Analysis - Mathematical Foundations. Springer Verlag, Heidelberg 1999; ISBN 3540627715
2. L. C. Paulson: Isabelle: A Generic Theorem Prover. Springer, 1994. LNCS 828. In P. Odifreddi, editor, Logic and Computer Science, pages 361-386. Academic Press, 1990.
3. T. Nipkow, L. C. Paulson, M. Wenzel: A Proof Assistant for Higher-Order Logic. Springer LNCS 2283, 2002.
4. L. C. Paulson: The Isabelle Reference Manual. `http://isabelle.in.tum.de/doc/ref.pdf`
5. T. Nipkow, L. C. Paulson, M. Wenzel: Isabelle's Logics: HOL. `http://isabelle.in.tum.de/doc/logics-HOL.pdf`
6. B. Ganter, R. Wille: Applied Lattice Theory: Formal Concept Analysis, 1997. `http://www.math.tu-dresden.de/~ganter/concept.ps`
7. M. Wenzel: Isabelle/Isar Reference Manual. `http://isabelle.in.tum.de/doc/isar-ref.pdf`.
8. M. Wenzel: Isabelle/Isar - a versatile environment for human-readable formal proof documents. PhD thesis, Institut für Informatik, Technische Universität München, 2002. `http://tumb1.biblio.tu-munchen.de/publ/dis/in/2002/wenzel.html`
9. F. Kammueller: Theory Tarski, 1999. `http://isabelle.in.tum.de/library/HOL/ex/Tarski.html`
10. B.A. Davey and H.A. Priestley: Introduction to Lattices and Order second edition, Cambridge University Press, 2002; ISBN 0521784514.
11. B. Sertkaya: Proof of the Basic Theorem on Concept Lattices in Isabelle/HOL. M.Sc thesis, Department of Computer Engineering, Middle East Technical University, Ankara, Turkey, 2003. `http://wwwtcs.inf.tu-dresden.de/~sertkaya/ms_thesis`
12. P. Rudnicki: An Overview of the Mizar Project. `http://mizar.org/project/bibliography.html`
13. C. Schwarzweller: Mizar Formalization of Concept Lattices, Mechanized Mathematics and its Applications, vol 1, 2000.
14. U. Krohn, N. J. Davies and R. Weeks: Concept Lattices for Knowledge Management, BT Technology Journal Vol 17, 1999.
15. R. Wille: Concept Lattices and Conceptual Knowledge Systems, Computers & Mathematics with Applications, 1992.
16. Y. Park: Software Retrieval by Samples Using Concept Analysis, Journal of Systems and Software, vol 1, 2000.
17. S. Berghofer: Program extraction in simply-typed higher order logic, LNCS 2646, 2002.
18. F. Puitg and J.F. Dufourd: Formalizing mathematics in higher-order logic: A case study in geometric modelling, Theoretical Computer Science, vol. 234, 2000.