A Path-Oriented Approach to Hierarchical Concept Structures

Marko Junkkari and Marko Niinimäki
University of Tampere
Department of Computer Science
April 14, 1998

Abstract: There are many principles to present a hierarchy of knowledge units i.e. concepts. A concept hierarchy exists on a set of concepts and it is based on a partial order relation. The set of concepts and the relation among these form a concept system. The purpose of this paper is to present a method and functions for inspecting the structure of a concept system and to provide a classification of typical structures.

Understanding the structure of a concept system is vital when one considers the utilisation of a concept system. The most common concept structures include trees, different lattices and semilattices. In order to utilise the operations for any type of a structure, it is important to recognise the structure. Our contribution here is to present a set of functions to recognise typical structures and to form a principle for classification of hierarchical concept structures. Here, the concept structures are represented on the basis of set theory, which is a well-known and established formalism. We introduce a path-oriented method that enables accurate and clear consideration of different structures.

1. Introduction

In artificial intelligence, knowledge representation and database design, many approaches to manage a hierarchical structure of "knowledge units" (classes, structured objects, concepts) have been proposed [Rich and Knight, 1991; Mac Randal, 1989; Ringland, 1989; Elmasri and Navathe, 1994]. Here, we assume that these knowledge units are concepts and we consider different types for hierarchical structures of concepts. We concentrate on hierarchical structures that concepts can form based on a binary relation in a given, finite set of concepts. Here we are interested in neither the ontological level on which concepts exist, nor in their semantics, nor in methods of discovering concepts.

For example Whythe [1969] sees that the use of hierarchical ordering must be as old as human thought itself and that it can be applied to many fields. One of these fields is the following category: "logical types, concepts, principles, information, quantities and abstractions of many sorts". In computer science, there has been much research on the field of hierarchical concept structures. In their historical work, Smith and Smith [1977] have presented principles of structures, based on generic-specific distinction. This type of hierarchies are known as subsumption or IS-A hierarchies. Their structure can be for example a tree or a lattice [Nilsson, 1994]. However, there are many other principles, too, on the basis of which one can form a hierarchy in a set of concepts [Nuopponen, 1994]. Here we discuss some structures more thoroughly and provide a categorisation of structures and functions for checking the structure at hand.

We assume that concepts are abstract, discrete units of knowledge [Junkkari and Niinimäki, 1998; see also Sowa, 1984 p.73]. There are many conceptions about the basic containment relation on the set of concepts. In this paper, we take the binary concept relation C-rel for representing this relation. It is considered irreflexive, non-transitive and antisymmetric, and moreover, the transitive closure of C-rel is antisymmetric. C-rel reflects partial order relation that may be either weak partial order relation (reflexive, transitive, antisymmetric) or strict partial order relation (irreflexive, transitive, antisymmetric). Now we assume that C-rel covers weak partial order containment relations, for example concept subsumption [Woods, 1991], IS-A relation [Brachman, 1983] and intensional containment [Kauppi, 1967; Palomäki, 1994]. There are also part-whole relations [Winston et al, 1987; Padgham and Lambrix, 1994; Sattler, 1995; Artale et al., 1996] that are either weak partial order relations or strict partial order relations. We assume that C-rel can also be applied to them. We consider only finite concept relations and the relation C-rel is on the finite concept set C-set.

Since C-rel reflects a partial order relation, it means that the structure imposed by this relation on a set of concepts is some sort of hierarchical ordering. The weakest type of structure will be called only a hierarchy. Sowa's [1984 p.80] type hierarchy is an example of this. In a type hierarchy, any two concepts (types) can have common subtypes and supertypes. If we restrict the structure so that any two concepts necessarily have a common supertype but they have not any common subtype, the structure is a tree. Von Linné's "system of organization for living organisms” is an example of a tree [Mayr, 1958][1].

Trees, lattices and semilattices are stronger structures than a hierarchy (that is, they are proper subsets of a set of hierarchies). Pedersen [1994] has used lattices for representing concepts and relationships in an information storage and retrieval system. Palomäki [1994] has pointed out that one axiomatisation of Kauppi's [1967] concept theory is a Boolean semilattice.

In several concept systems, there exists a concept that represents all individuals in the given Universe of Discourse. In Description Logics (DL) and Classic [Borgida et. al., 1989; Borgida, 1995], this concept is called THING and it subsumes all other concepts. In Kauppi's [1967] concept theory the corresponding concept is called the general concept which is intensionally contained in all concepts. In our illustrations, concepts in which no other concepts are contained are depicted below all the other concepts, like in figure 1. In our graphical representation, a concept on an upper level contains a concept on a lower level connected to it. In our formal presentation, for example the ordered concept pair <MATERIAL, THING> inheres in the relation C-rel. In this case this is interpreted: “MATERIAL IS A THING” or “THING subsumes MATERIAL”.

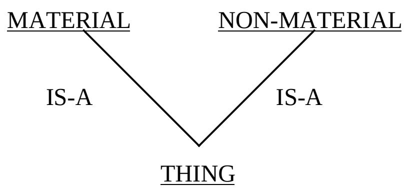

Figure 1. A graphical presentation of IS-A relation.

Due to the IS-A relation, it is natural to have one concept that is contained in every concept. If the containment relation is applied to other sorts of containment (e.g. whole-part) we need to employ another graphical method, the one that is presented in figure 2. There, the most abstract concept is depicted above all the other concepts [Kangassalo, 1983a; Kangassalo, 1993]. This concept corresponds to the subject of conceptual modelling, like a company. The concept of company contains all the other concepts of the Universe of

[1] According to Mayr [1958], Linné attempted to establish groups distinguished be the following ways:
(a) Having the greatest possible number of common characteristics.
(b) Having a key character, a diagnostic character that would automatically guarantee correct idenfication.

Discourse. A basic concept is a concept that contains no other concepts than itself. Basic concepts are depicted below all other concepts [Kangassalo, 1983b].

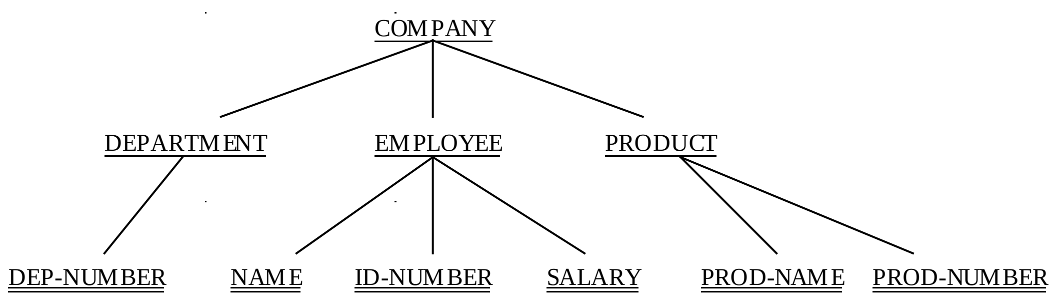

Figure 2. A concept hierarchy with one top-concept.

We call bottom-concepts[2] those concepts that do not contain other concepts. We call a top-concept such a concept, which is not contained in any other concept. In figure 2, there is one top-concept and several bottom-concepts.

By combining the visualization principles used in the figures 1 and 2, we end up with a structure that has exactly one most abstract concept and exactly one general concept. If we want to impose some further constraints on the relation between concepts, the resulting structure on the set of concepts is a lattice.

In this paper, we give an approximate classification for concept structures based on four categories. This is done by inspecting the number of top-concepts and bottom-concepts. Inside some categories, we can make subcategories (stronger structures). We also present methods for checking the category to which a given concept system belongs. This is done by means of a path-oriented approach [see Niemi and Järvelin, 1992].

In section 2, we present basic notations needed in this paper. Section 3 discusses the relation C-rel on the set of concepts. We define two functions: *top_concept_set* and *bottom_concept_set* for collecting on C-rel top-concepts and bottom-concepts respectively. These sets and paths top-concepts to bottom-concepts form the basis of our categories. In section 4, we introduce the basic types of connections between two concepts. This is generalised by using a path oriented definition method. Section 5 discusses connected and disconnected concept systems. The categories are presented in section 6, where we also construct functions for checking the category for a given concept system. Finally, in section 7 we give a simple analysis of relationships among concepts in various concept structures. Moreover, we maintain that our methods are suitable for analysis of more complicated situations, i.e. a concept structure is a model from several Universes of Discourse.

2. Mathematical background

In this paper, some fields of mathematics are prerequisite. Our formalism is based on set theory, more specifically on formalism developed by Niemi and Järvelin [1992]. In this notation, a finite n-tuple is denoted between angle brackets, for example <a,b,c>. A two-place tuple is also called an ordered pair. A set of ordered pairs is always a binary relation. The empty tuple is denoted by <>. The tuple set of a set S is denoted by T(S). A tuple is an element of T(S) if all elements of the tuple are different and they are members in the set S. The tuple set T(S) is composed of all permutations of the power set P(S) which are presented as tuples. For example, if S ={a, b, c} then the T(S) is {<>, <a>, <b>, <c>, <a,b>, <b,a>, <a,c>, <c,a>, < b,c>, <c,b>, <a,b,c>, <a,c,b>, <b,a,c>, <b,c,a>, <c,a,b>, <c,b,a>}.

Functions are described normally so that also their signatures are given explicitly. Generally a signature for function *f* is denoted by *f*: S1 $\rightarrow$ S2. S1 defines a set of values to which the function *f* can be applied and S2 defines the set to which values yielded by f belong.

In this paper, we introduce a partial order relation that does not include reflexive and transitive relationships explicitly. This kind of a relation can be deduced from a partial order on a finite set. Formally:

- If A is a finite set and R is a strict partial order relation on it then the predecessor of an element b is the element a provided <a,b> $\in$ R and there is no element x $\in$ A such that <a,x> $\in$ R and <x,b> $\in$ R. The successor of an element b is the element c provided <b,c> $\in$ R and there is no element x $\in$ A such that <b,x> $\in$ R and <x,c> $\in$ R.
- The subset of R that contains only predecessor-successor -pairs is called the immediate subset of R. For example, if R is called a containment relation, then its immediate subset is called the immediate containment relation.

In addition to set theory, we need basics of lattice theory and algebra. We discuss axiomatised systems that have several operations. An operation is a relation. For example, the operation of least upper bound (lub) is a three-placed relation expressed by a lub b = c. We present some basic properties of algebra operations:

- An operation o is idempotent if x o x = x for any x $\in$ A.
- An operation o is commutative if x o y = y o x for any x, y $\in$ A.

[2] For example Borgida [1995] would call our unique bottom-concept TOP-CONCEPT.

- An operation o is associative if (x o y) o z = x o (y o z) for any x, y, z $\in$ A.

In addition to these definitions, we apply some basic notions of graph theory. We assume that the reader is familiar with them.

3. Basic relation and concept system.

C-set is the finite set of concepts related to a specific concept system. In each concept system, there is a containment relation C-rel on the set C-set. This relation is a binary relation that corresponds to immediate containment. As an example, let us take a concept set C-set1 = {c1, c2, c3, c4, c5, c6, c7, c8} and a relation on it, C-rel1 = {<c1,c2>, <c1,c8>, <c2,c6>, <c3,c4>, <c3,c5>, <c4,c6>, <c4,c7>, <c5,c7>, <c5,c8>}. This is presented in the figure 3 below.

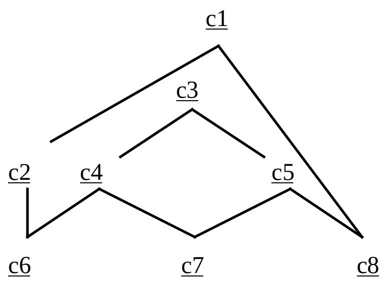

Figure 3: C-rel1.

Any C-rel describes a concept system and transitive and reflexive relationships among concepts can be defined when necessary. A simple way to do this is by using closures [Niemi and Järvelin, 1992; Junkkari and Niinimäki, 1998] or a path-oriented approach [Niemi and Järvelin, 1992].

We define a function *relation_on* to generate a C-rel-like substructure of the original C-rel from a given concept set C-set'.

relation_on: P(C-set) $\times$ P(C-set $\times$ C-set) $\rightarrow$ P(C-set $\times$ C-set)
relation_on(C-set', C-rel) = {<x,y> | x $\in$ C-set' $\wedge$ y($\neq$x) $\in$ C-set': <x,y> $\in$ C-rel}

For example, in C-rel1, the function *relation_on*({c2,c4,c6,c7}, C-rel1) produces the relation {<c2,c6>, <c4,c6>, <c4,c7>}.

In any non-empty concept system, there is at least one concept, which does not contain any concepts in C-rel, and at least one concept, which is not contained in any concepts in C-rel. These two cases have a specific role in our system. Next, we define the functions that return the sets of these specific concepts. The function *bottom_concept_set* returns the set of such concepts, which do not have successors in C-rel.

bottom_concept_set : $P(\text{C-set} \times \text{C-set}) \rightarrow P(\text{C-set})$

bottom_concept_set(C-rel) = $\{x \mid x \in \text{C-set} \wedge \neg\exists y \in \text{C-set} : \langle x,y\rangle \in \text{C-rel}\}$

For example, the function *bottom_concept_set*(C-rel1) returns the set {c6, c7, c8}.

Respectively, the function *top_concept_set* computes the set of all concepts that do not have any predecessors in C-rel.

top_concept_set : $P(\text{C-set} \times \text{C-set}) \rightarrow P(\text{C-set})$

top_concept_set (C-rel)= $\{x \mid x \in \text{C-set} \wedge \neg\exists y \in \text{C-set} : \langle y,x\rangle \in \text{C-rel}\}$

Clearly, *top_concept_set*(C-rel1) returns the concept set {c1, c3}.

Furthermore, we need the operation in terms of which we can consider indirect or immediate containments between two concepts. For this purpose we define the function *path_set* which returns the set of paths from a concept a to a concept b in the concept system at hand. It generates all possible paths for connecting two concepts. The function takes three arguments: two concepts, a and b, and a containment relation C-rel. On the basis of ordered pairs in C-rel, we construct the paths from a to b (see the operation 15 in [Niemi and Järvelin, 1992][3]).

path_set: $\text{C-set} \times \text{C-set} \times P(\text{C-set} \times \text{C-set}) \rightarrow P(T(\text{C-set}))$

path_set(a, b, C-rel)=

$$\begin{cases} \{< a_1,..,a_n > \mid < a_1,..,a_n > \in \ T(\text{C}-\text{set}) : a_1 = a \ \wedge a_n = b \wedge \forall i \in \{1,..,n-1\} : \\ \qquad\qquad < a_1, a_{i+1} > \in \ \text{C-rel}\}, \text{ if } a \neq b \\ \{< a >\}, \text{ if } a = b \end{cases}$$

Now in the example in fig. 3 *path_set*(c1, c6, C-rel1) produces the set {<c1,c2,c6>} and respectively, *path_set*(c3, c7, C-rel1) = {<c3,c4,c7>, <c3,c5,c7>}. For example *path_set*(c1, c5, C-rel1) and *path_set*(c8, c1, C-rel1) return an empty set.

By using the function *path_set* we can consider all possible paths between any two concepts in a concept system. Here we are especially interested in paths from top-concepts to bottom-concepts.

4. Basic structures

We take five situations for considering connections between two concepts. First we introduce these five structures in a simple form and then we generalise these situations by using path-oriented approach. In section 7, we use these situations to analyse different concept structures.

[3]Niemi and Järvelin have taken also the relation base as an argument. Our assumption in the operation is that, if a and b are identical, the function returns {<a>}.

The immediate containment between two concepts a and b has two alternatives: a contains b, denoted by <a,b> ∈ C-rel, or b contains a, denoted by <b,a> ∈ C-rel. If we take a third concept c then we can associate concepts a and b via it. We say that concepts a and b are immediately *top-connected*, if there is a concept c that contains immediately both the concepts a and b, i.e. <c,a> and <c,b> belong to C-rel. Respectively, we say that concepts are immediately *bottom-connected*, if there is a concept d such that <a,d> ∈ C-rel and <b,d> ∈ C-rel. Moreover, concepts are immediately *cross-connected* if they are immediately top-connected and bottom-connected. The basic structures are presented in figure 4.

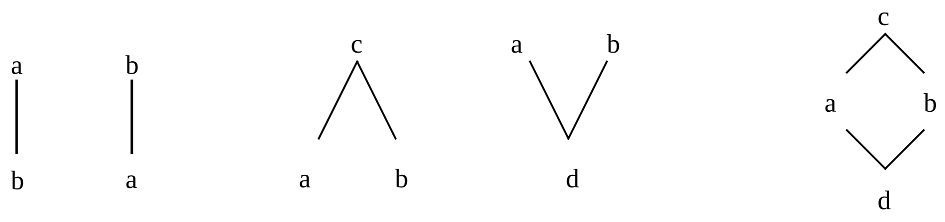

Figure 4: Basic structures: contains, is contained, top-connected, bottom-connected and cross-connected.

We generalise these basic structures to take into account indirect (transitive) relationships among concepts in C-rel.

The first generalisation concerns contains relation. The Boolean function *contains*(a, b, C-rel) returns true, if in C-rel there is at least one path from the concept a to the concept b. This means that a contains b immediately or indirectly (transitively).

contains: C-set × C-set × P(C-set × C-set) → {false, true}
contains(a, b, C-rel) =

$$\begin{cases}\text{true, if } \mathit{path\_set}(\text{a, b, C - rel}) \neq \{\} \\ \text{false, otherwise}\end{cases}$$

The converse for the *contains* function is *is_contained* function. The function *is_contained*(a, b, C-rel) yields true, if and only if the concept a is contained in the concept b immediately or indirectly.

is_contained: C-set × C-set × P(C-set × C-set) → {false, true}
is_contained(a, b, C-rel) =

$$\begin{cases}\text{true, if } \mathit{contains}(\text{b, a, C - rel}) \\ \text{false, otherwise}\end{cases}$$

We get all those concepts which the concept a contains by applying the function *contains_set*(a, C-rel). Respectively, the function *is_contained_set*(a, C-rel) returns all those concepts which contain the concept a. Next we define these functions.

contains_set: C-set $\times$ P(C-set $\times$ C-set) $\rightarrow$ P(C-set)
contains_set(a, C-rel) = $\{x \mid x \in$ C-set $\wedge$ *contains*(a, x, C-rel)$\}$

is_contained_set: C-set $\times$ P(C-set $\times$ C-set) $\rightarrow$ P(C-set)
is_contained_set(a, C-rel) = $\{x \mid x \in$ C-set $\wedge$ *is_contained*(x, a, C-rel)$\}$

If a and b are top-connected with each other then there is such concept x from which there is at least one path both to the concept a and to the concept b. It is easy to see that in C-rel1 every concept is top-connected with all the other concepts. The function *top_connected*(a, b, C-rel) is true if two concepts a and b are top-connected with each other.

top_connected : C-set $\times$ C-set $\times$ P(C-set $\times$ C-set) $\rightarrow$ {false, true}
top_connected(a, b, C-rel)=

$$\begin{cases} \text{true, if } \exists x \in \text{C-set} : \textit{contains}(x, a, \text{C-rel}) \wedge \textit{contains}(x, b, \text{C-rel}) \\ \text{false, otherwise} \end{cases}$$

We say that two concepts a and b are bottom-connected, if and only if there is a concept x which is contained immediately or indirectly in both the concepts a and b. In our path-oriented formalism, this means that there exists a path both from the concept a and the concept b to the concept x. The function *bottom_connected*(a, b, C-rel) is true if two concepts a and b are bottom-connected with each other

bottom_connected : C-set $\times$ C-set $\times$ P(C-set $\times$ C-set) $\rightarrow$ {false, true}
bottom_connected(a, b, C-rel)

$$\begin{cases} \text{true, if } \exists x \in \text{C-set} : \textit{contains}(a, x, \text{C-rel}) \wedge \textit{contains}(b, x, \text{C-rel}) \vee a = b \\ \text{false, otherwise} \end{cases}$$

By combination of top-connected and bottom-connected concepts, we get cross-connected concepts. The function *cross_connected*(a,b,C-rel) is true, if and only if the concepts a and b are both top-connected and bottom-connected.

cross_connected: C-set $\times$ C-set $\times$ P(C-set $\times$ C-set) $\rightarrow$ Boolean
cross_connected(a, b, C-rel)

$$\begin{cases} \text{true, if } \mathit{top\_connected}(a, b, C\text{-rel}) \wedge \mathit{bottom\_connected}(a, b, C\text{-rel}) \\ \text{false, otherwise} \end{cases}$$

5. Connected and partial structures

In this section we consider whether a concept system is connected or disconnected. If the system is connected then some connection for any two concepts has to be. This connection can be direct or indirect through one or several concepts. For example in figure 5, all the concepts are connected with each other.

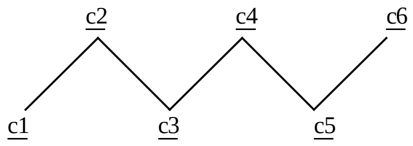

Figure 5. A connected structure.

We define the following functions in order to find connected substructures in a concept system. The recursive function *connected_concepts*(s, S, C-rel), where S = *top_concept_set*(C-rel), returns a set of concepts that are in some way connected with the given top-concept s, i.e. s ∈ S.

connected_concepts: C-set × P(C-set) × P(C-set × C-set) → P(C-set)
connected_concepts(s, S, C-rel) =

$$A \cup \bigcup_{x(\neq s) \in T} \mathit{connected\_concepts}(x, S - \{s\}, C\text{-rel})$$

$$\text{where } A = \mathit{contains\_set}(s, C\text{-rel}) \wedge$$

$$T = \{s' \mid \exists < s, \ldots b >, \exists < s', \ldots, b > \in \mathit{path\_set}(a, b) : a \in S \wedge$$

$$b \in \mathit{bottom\_concept\_set}(C\text{-rel})\}$$

The function *substructures* return a set of distinct sets of concepts.

substructures: P(C-set × C-set) → P(P(C-set))
substructures(C-rel) = {X | ∀y ∈ *top_concept_set*(C-rel):
X = *connected_concepts*(y, *top_concept_set*(C-rel), C-rel)}

Now if a concept system is connected, there is only one set in the set that the function returns. Using this we can define the function that checks if a concept system is connected. If a concept system is not connected then it is disconnected, i.e. the function yields false.

connected: P(C-set × C-set) → {false, true}
connected(C-rel) =

$$\begin{cases} \text{true, if } |\, substructures(\text{C - rel}) \,| = 1 \\ \text{false, otherwise} \end{cases}$$

6. Approximate classification for concept structures

The first principle for our classification is the number of top and bottom-concepts in a concept system. We are especially interested in whether the number of top-concepts or bottom-concepts equals to one. In the strongest category, it holds that there is exactly one top-concept and bottom-concept. In this case, we call the concept system a *closed hierarchy*. A special case of this is a lattice (see section 6.1). Since we only consider finite concept systems, the lattice is always bounded [see Birkhoff, 1993].

If either the number of top-concepts or the number of bottom-concepts equals to one, we call the concept system a *semi-closed hierarchy*. The two cases of these are a *top-closed hierarchy* and a *bottom-closed hierarchy*. The functions in both cases are very much alike, and we consider thoroughly only top-closed hierarchy in section 6.2. The special cases of a semi-closed hierarchy are tree and semilattice.

If there are more than one top-concepts and more than one bottom-concepts in the concept system, we call the concept system an *open hierarchy*. This is the only case where it is reasonable to inspect whether the concept system is disconnected. If the concept system is disconnected, we can inspect each of the substructures. In this way we can find out if the concept system is for example a collection of lattices.

	|*top_concept_set*(C-rel)| = 1	|*top_concept_set*(C-rel)| > 1
|*bottom_concept_set*(C-rel)| =1	Closed hierarchy -connected -lattice	Bottom-closed hierarchy -connected -semilattice -tree
|*bottom_concept_set*(C-rel)| > 1	Top-closed hierarchy -connected -semilattice -tree	Open hierarchy 1. connected 2. disconnected -collection

Figure 6. Classification of structures.

6.1. Closed hierarchy

A closed hierarchy has exactly one top-concept and one bottom-concept. The function *closed_hierarchy* returns true, if these conditions are satisfied.

closed_hierarchy: P(C-set $\times$ C-set) $\rightarrow$ {false, true}
closed_hierarchy(C-rel)=

$$\begin{cases} \text{true, if } |\,top\_concept\_set(\text{C-rel})\,| = 1 \wedge |\,bottom\_concept\_set(\text{C-set})\,| = 1 \\ \text{false, otherwise} \end{cases}$$

In figure 7, all the structures are closed hierarchies. Only (b) and (c) are lattices. First we introduce functions to inspect whether the concept system is a lattice.

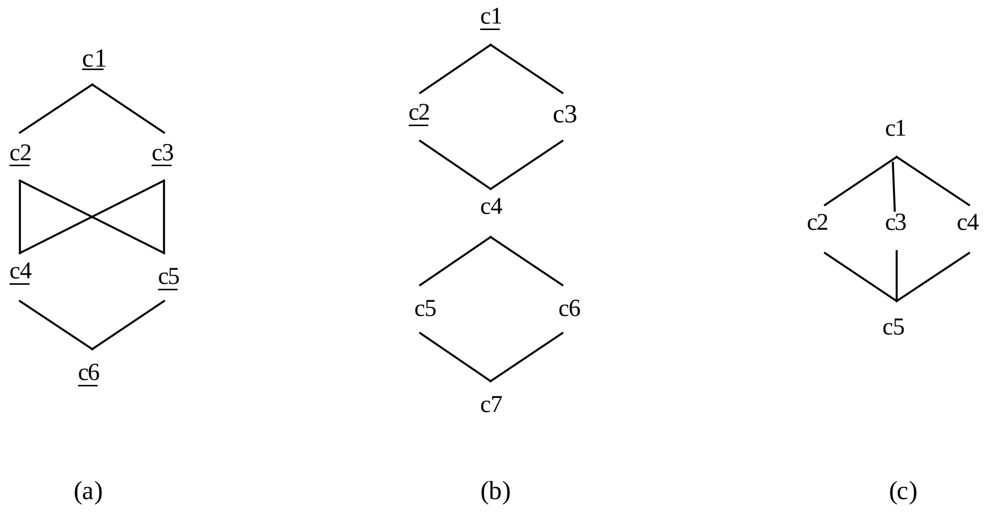

Figure 7. Closed hierarchies.

In a concept system, the *lower bounds* of two concepts, a and b, are concepts that are contained in both a and b. The *greatest lower bound* (glb) for concepts a and b is the concept that is nearest to a and b in the set of their *lower bounds*. If it exists, it is unique. The *least upper bound* (lub) is defined respectively.

In a lattice, any two concepts have the greatest lower bound (glb) and the least upper bound (lub). In order to check the unique greatest lower bound of two concepts a and b we inspect the sets that the function *contains_set* returns for both a and b. We study the substructure of C-rel restricted by the intersection of these sets. If the constructed substructure has exactly one top-concept then a and b have one greatest lower bound. This is done in the function *check_glb*. The check is done for all concepts in function *check_glbs*. The functions *check_lub* and *check_lubs* are defined in an analogous manner.

check_glb: C-set $\times$ C-set $\times$ P(C-set $\times$ C-set) $\rightarrow$ {false, true}
check_glb(a, b, C-rel) =

$$\begin{cases} \text{true, if } |\,top\_concept\_set(relation\_on(contains\_set(\text{a, C-rel}) \cap \\ \quad contains\_set(\text{b, C-rel})))\,| = 1 \vee contains(\text{a, b, C-rel}) \vee contains(\text{b, a, C-rel}) \\ \text{false, otherwise} \end{cases}$$

check_glbs: P(C-set $\times$ C-set) $\rightarrow$ {false, true}
check_glbs(C-rel)=

$$\begin{cases} \text{true, if } \forall x \forall y (\neq x) \in \text{C-set}: \textit{check\_glb}(x, y) \\ \text{false, otherwise} \end{cases}$$

check_lub: C-set $\times$ C-set $\times$ P(C-set $\times$ C-set) $\rightarrow$ {false, true}

$$\begin{cases} \text{true, if } |\textit{bottom\_concept\_set}(\textit{relation\_on}(\textit{is\_contained\_set}(a, \text{C-rel}) \cap \\ \quad \textit{is\_contained\_set}(b, \text{C-rel})))| = 1 \vee \textit{contains}(a, b, \text{C-rel}) \vee \textit{contains}(b, a, \text{C-rel}) \\ \text{false, otherwise} \end{cases}$$

check_lub(a, b, C-rel)=

check_lubs: P(C-set $\times$ C-set) $\rightarrow$ {false, true}

$$\begin{cases} \text{true, if } \forall x \forall y (\neq x) \in \text{C-set}: \textit{check\_lub}(x, y) \\ \text{false, otherwise} \end{cases}$$

check_lubs(C-rel)=

A structure is a lattice if there is a unique greatest lower bound and a unique least upper bound for any two concepts. The function *check_lattice* inspects if C-set is a lattice.

check_lattice: P(C-set $\times$ C-set) $\rightarrow$ {false, true}

$$\begin{cases} \text{true, if } \textit{check\_glbs}(\text{C-rel}) \wedge \textit{check\_lubs}(\text{C-rel}) \\ \text{false, otherwise} \end{cases}$$

check_lattice(C-rel) =

For example, in the figure 8(a), the structure is not a lattice, since there is no unique least upper bound for the concepts c4 and c5 and there is no unique greatest lower bound for concepts c2 and c3.

A lattice can be distributive or complete [Birkhoff, 1993], but we do not include function for checking these special lattices in this paper.

6.2. Semi-closed hierarchy

A semi-closed hierarchy can be top-closed or bottom-closed. We inspect the top-closed hierarchy in more detail. The function *top_closed_hierarchy* inspects whether the hierarchy is top-closed.

top_closed_hierarchy: P(C-set $\times$ C-set) $\rightarrow$ {false, true}
top_closed_hierarchy(C-rel)=

$$\begin{cases} \text{true, if } |\textit{top\_concept\_set}(\text{C-rel})| = 1 \\ \text{false, otherwise} \end{cases}$$

Analogously, one can inspect whether the concept system is a bottom-closed hierarchy. This inspection can be based on the number of bottom concepts.

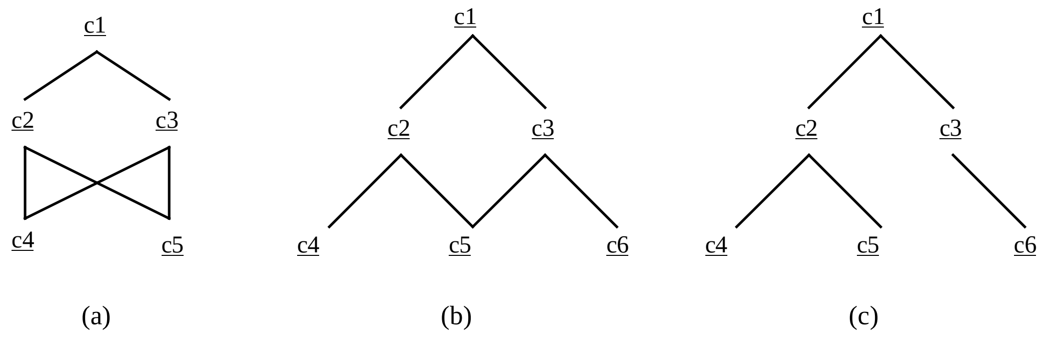

Figure 8: Some top-closed hierarchies.

In a semi-closed hierarchy, C-set can be a semilattice. A semilattice is a system with a one idempotent, commutative and associative operation [Birkhoff, 1993]. It is either the glb or the lub operation.

$check\_semilattice$: P(C-set $\times$ C-set) $\rightarrow$ {false, true}
$check\_semilattice$(C-rel) =

$$\begin{cases} \text{true, if } \mathit{check\_glbs}(\text{C-rel}) \vee \mathit{check\_lubs}(\text{C-rel}) \\ \text{false, otherwise} \end{cases}$$

If we want to study for example top-closed semilattices, we only check the lub operation. In figure 8, (b) and (c) are top-closed semilattices.

If C-rel is a top-closed hierarchy, it can be a directed tree. If a top-closed hierarchy is a tree, then the only top-concept is called the root of the tree. We can use the function *check_tc_tree*(C-rel) for checking that C-rel is a directed tree. A directed graph is a directed tree, if and only if it has a root from which there is a unique (directed) path to every vertex (concept) [Even, 1979]. It can be proved that a top-closed hierarchy is a tree if there is a unique path from the top-concept to every bottom-concept. The function *check_tc_tree* gives the value true, if there are these unique paths in the relation C-rel.

$check\_tc\_tree$: P(C-set $\times$ C-set) $\rightarrow$ {false, true}
$check\_tc\_tree$(C-rel)=

$$\begin{cases} \text{true, if } \mathit{top\_closed\_hierarchy}(\text{C-rel}) \wedge \forall x \in \mathit{bottom\_concept\_set}(\text{C-rel}): \\ \qquad |\mathit{path\_set}(a, x, \text{C-rel})| = 1 \text{ where } a \in \mathit{top\_concept\_set}(\text{C-rel}) \\ \text{false, otherwise} \end{cases}$$

If the concept system is a bottom-closed hierarchy, it can be an inverted tree, which is a tree based on the inverse relation of containment. The function for checking this is analogous to the above definition.

6.3. Open hierarchy

The last situation of our rudimentary classification for concept structures is an open hierarchy. In an open hierarchy, there are more than one top-concepts and more than one bottom-concepts. It is inspected as follows:

open_hierarchy: P(C-set $\times$ C-set) $\rightarrow$ {false, true}

$$\begin{cases}\text{true, if } |\, \mathit{top\_concept\_set}(\text{C-rel})\,| > 1 \wedge |\, \mathit{bottom\_concept\_set}(\text{C-rel})\,| > 1 \\ \text{false, otherwise}\end{cases}$$

open_hierarchy(C-rel)=

Unlike other categories, an open hierarchy can be connected or disconnected. If there are any connections between any two concepts in a concept system, then the open hierarchy is connected. The function *open_connected_hierarchy* returns true if C-rel is connected and the concept system is an open hierarchy:

open_connected_hierarchy: P(C-set $\times$ C-set) $\rightarrow$ {false, true}
open_connected_hierarchy(C-rel) =

$$\begin{cases}\text{true, if } \mathit{connected}(\text{C-rel}) \wedge \mathit{open\_hierarchy}(\text{C-rel}) \\ \text{false, otherwise}\end{cases}$$

In an open hierarchy, we are interested in finding the components of C-rel that are sets of concepts that are connected to each other. In section 5, we defined the function *substructures* that finds the connected components of a concept system. We can check if all the components correspond to the given structure. As an example, the next function checks if the components are lattices.

collection_of_lattices: P(C-set $\times$ C-set) $\rightarrow$ {false, true}
collection_of_lattices(C-rel) =

$$\begin{cases}\text{true, if } \mathit{open\_hierarchy}(\text{C-rel}) \wedge \forall \text{X} : \text{X} \in \mathit{substructures}(\text{C-rel}) \wedge \mathit{check\_lattice}(\text{X}) \\ \text{false, otherwise}\end{cases}$$

It is trivial to construct similar functions that inspect if the concept system is a collection of trees, a collection of inverted trees etc.

7. An analysis of concept structures

The introduction to our categorisation has been based on the number of top-concepts and bottom-concepts. The closed hierarchy is a structure, where there is exactly one top-concept and one bottom-concept. Considering relations that were presented in the section 4, we can notice that every two concepts in this kind of system are cross-connected. Moreover, we can check if the concept structure is a lattice.

The top-closed hierarchy is a concept hierarchy where there is one top-concept and several bottom-concepts. In this kind of a structure any two concepts are top-connected. There can be some bottom-connected concepts with each other. In the case of that the structure is a semilattice. If any two concepts are not bottom-connected then the concept structure is a tree.

Respectively, as above, in a bottom-closed hierarchy any two concepts are bottom-connected. In this case the structure is a tree if there are no two such concepts, which are top-connected. If the structure is a semilattice then there can be some concept pairs that are top-connected.

The last major category in our discussion was open hierarchy. Now there are several top-concepts and bottom-concepts. In an open hierarchy, there has to be at least two concepts that are not bottom-connected and two such concepts that are not top-connected. The open hierarchy can be connected or disconnected. A disconnected structure can be a collection of other structures. Therefore, we can check if every substructure is, for example, a lattice or a tree and we can consider collections of stronger structures.

The question of conceptual collection becomes advantageous, if we consider information models from several Universes of Discourse. If we assume that a concept lattice is characteristic of one Universe of Discourse, then naturally a collection of isolated concept systems is a collection of concept lattices [Kangassalo, 1997].

The situation can be such that concept lattices are not totally isolated, but there are some common concepts. Finding these concepts forms a basis to knowledge sharing from different directions to use common information resource. Using our systems, it is easy to present formally, how to find these concepts.

As an example, we present a function that produces the relation, which is the common structure of two concept lattices. Let us assume two concept lattices C-rel1 and C-rel2 that are relations on the set C-set.

sharing_lattice: $P(\text{C-set} \times \text{C-set}) \times P(\text{C-set} \times \text{C-set}) \rightarrow P(\text{C-set} \times \text{C-set})$
sharing_lattice (C-rel1, C-rel2) =

$$\begin{aligned}&\textit{relation\_on}\ (\textit{contains\_set}(x, \text{C-rel}) \cap \textit{contains\_set}(y, \text{C-rel})\\ &\qquad \cap \textit{is\_contained\_set}(z, \text{C-rel}) \cap \textit{is\_contained\_set}(w, \text{C-rel}))\\ &\quad \text{where } x \in \textit{top\_concepts}(\text{C-rel1}) \wedge y \in \textit{top\_concepts}(\text{C-rel2}) \wedge\\ &\qquad z \in \textit{bottom\_concepts}(\text{C-rel1}) \wedge w \in \textit{bottom\_concepts}(\text{C-rel2})\end{aligned}$$

The result of the function *sharing_lattice* is a concept structure and in this paper, we have presented the functions for checking the form of this structure. In the same way, it is possible to construct operations that produce the common substructure for any two concept systems. In addition, it is possible to enlarge operations to sets of structures so that we can consider several concept systems.

The consideration can be also applied so that for example in an open hierarchy we choose one top-concept and then we consider the substructure that the concept determines. Therefore, we can also make concept analysis under one concept system.

8. Summary

In this paper, we have discussed various concept structures based on the containment relation. Our formalisation approach can be characterised path-oriented which, to our knowledge, is a novel approach in concept theory. In this paper, we presented a categorisation for concept structures.

The containment relation in our study was partial order and we have noticed that several concept structures can be presented based on it. In general, this means that concept structures of this type can be presented as a directed acyclic graph. There are also stronger structures such as lattices, semilattices and trees and we have presented how to recognise them. Moreover, we have presented how to recognise a collection of similar structures. In addition, we have argued that with this kind of system is possible to analyse common parts in different concept systems.

The definitions and functions given in this paper provide many possibilities for conceptual modelling. For instance, if there are two concept systems, the functions reveal their structure. More functions can be designed to discover the part that is common for the concept systems.

References

[Artale et al., 1996] Alessendro Artale, Enrico Franconi, Nicola Guarino and Luca Pazzi, Part-Whole Relations in Object-Centered Formalisms: an Overview, Data and Knowledge Engineering. Vol. 20, Issue 3, Elsevier Science Publishers, 1996. Pp 347-383.

[Birkhoff, 1993 (1940)] Garrett Birkhoff, Lattice Theory. American Mathematical Society: Colloquium Publications, Vol. 25, Third edition, First published 1940, Providence, Rhode Island, 1993.

[Borgida et al, 1989] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness and Lori Alperin Resnick, Classic: A structural data model for objects, ACM Sigmod Record. Vol. 18, No. 2, ACM Press, 1989. Pp. 58-67.

[Borgida, 1995] Alexander Borgida, Description logics in data management, IEEE transactions on knowledge and data engineering. Vol. 7, No. 1, 1995. Pp. 671-682.

[Brachman et al, 1991] Ronald J. Brachman, Deborah L. McGuinness, Peter F. Patel-Schneider, Lori Alperin Resnick and Alexander Borgida, Living with Classic: When and how to use a KL-ONE-like language, in J. F. Sowa (ed.), Principles of Semantic Networks. Morgan Kaufman Publishers, San Mateo, California, 1991. Pp. 401-455.

[Brachman, 1983] Ronald J. Brachman, What IS-A is and isn't: An analysis of taxonomic links in semantic networks, IEEE Computer. Vol. 16, No. 10, 1983. Pp. 30-36.

[Elmasri and Navathe, 1994] Ramez Elmasri and Shamkant Navathe, Fundamentals of Database Systems, Benjamin/Cummings Publishing Company, 1994.

[Even, 1979] Shimon Even, Graph Algorithms. Pitman, London, 1979.

[Junkkari and Niinimäki, 1998] Marko Junkkari and Marko Niinimäki, An Algebraic Approach to Kauppi's Concept Theory, Technical Report A-1998-1, Department of Computer Science, University of Tampere, Tampere, 1998.

[Kangassalo, 1983a] Hannu Kangassalo, Structring principles of conceptual schemas and conceptual models, in Janis A. Bubenko Jr (ed.) Information Modeling. Studentlitteratur, Lund, 1983. Pp 223-307.

[Kangassalo, 1983b] Hannu Kangassalo, CONCEPT D -A graphical formalism for representing concept structures, in Hannu Kangassalo (ed.), Second Scandinavian Research Seminar on Information Modelling and Database Management. Ser. B, Vol. 19, University of Tampere, Tampere, 1983. Pp. 167-198.

[Kangassalo, 1997] Hannu Kangassalo, personal communication, December 1997.

[Kangassalo, 1993] Hannu Kangassalo, COMIC: A system and methodology for conceptual modelling and information construction, Data and Knowledge Engineering 9. Elsevier Science Publishers, North-Holland, 1993. Pp. 287-319.

[Kauppi, 1967] Raili Kauppi, Einführung in die Theorie der Begriffssysteme. Acta Universitatis Tamperensis, Ser. A, Vol. 15, Tampereen yliopisto, Tampere, 1967.

[Mac Randal, 1989] Damian MacRandal, Semantic networks, in G.A. Ringland and D.A. Duce (eds.) Approaches to Knowledge Representation, John Wiley & Sons, New York, 1989.

[Mayr, 1958] Ernst Mayr: The Evolutionary Significance of Systematic Categories in Olov Hedberg (ed.) Systematics of To-Day, Uppsala Universitetets Årsskrift, 1958:6.

[Niemi and Järvelin, 1992] Timo Niemi and Kalervo Järvelin, Operation-oriented query language approach for recursive queries - Part 1: Functional definition, Information systems. Vol. 17 No. 1, Pergamon Press plc, 1992. Pp. 49-75.

[Nilsson, 1994] Jørgen Nilsson, An algebraic logic for concepts structures, in Hannu Jaakkola, Hannu Kangassalo, Tadahiro Kitahashi and Andras Markus (eds.), Information Modelling and Knowledge Bases V. Amsterdam, IOS Press, 1994. Pp. 75-84 .

[Nuopponen, 1994] Anita Nuopponen, Begreppssystem för terminologisk analys. (Concept Systems for Terminological Analysis), Doctoral Thesis, Acta Wasaensia, No 38, University of Vaasa, 1994.

[Padgham and Lambrix, 1994] Lin Padgham and Patrick Lambrix, A framework for part-of hierarchies in terminological logics, Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference(KR '94). Morgan Kaufman Publishers, Inc. San Francisco, 1994. Pp. 485-496.

[Palomäki, 1994] Jari Palomäki, *From Concepts to Concept Theory: Discoveries, Conections and Results.* Doctoral Thesis, Acta Universitatis Tamperensis, Ser. B, Vol. 416, Tampereen yliopisto, Tampere, 1994.

[Pedersen, 1994] Gert Schmeltz Pedersen: Relationship Lattices for Information Modelling, in H. Jaakkola et al (eds.) Information Modelling and Knowledge Bases V, IOS Press, Amsderdam, 1994.

[Rich and Knight, 1991] Elaine Rich and Kevin Knight, Artificial Intelligence. 2nd. ed., McGraw-Hill, 1991.

[Ringland, 1989] Structured object representation - Schemata and frames, in G.A. Ringland and D.A. Duce (eds.) Approaches to Knowledge Representation, John Wiley & Sons, New York, 1989.

[Sattler, 1995] Ulrike Sattler, A concept language for a engineering application with part-whole relations, in Alexander Borgida and Daniele Nardi (eds.), Proceedings of the International Workshop on Description Logics. Rome, 1995. Pp. 119-123.

[Smith and Smith, 1977] J. M. Smith and D. C. Smith, Database abstraction: Aggregation and Generalization, ACM Transaction on database systems. Vol. 12, No. 2, 1977. Pp. 105-133.

[Sowa, 1984] John F. Sowa, Conceptual Structures. Addison Wesley, 1984.

[Whyte, 1969] Lancelot Law Whyte, Structural hierarchies: A challenging class of physical and biological problems, in L. L. Whyte, Albert G. Wilson and Donna Wilson (eds.) Hierarchical Structures. Elsevier Publishing Company, New York, 1969.

[Winston et al, 1987] Morton E. Winston, Roger Chaffin and Douglas Herrmann, A taxonomy of part-whole relations, Cognitive science. Vol. 11, No. 4, 1987. Pp. 417-444.

[Woods, 1991] W. A. Woods, Understanding subsumption and taxonomy: A framework for progress, in J. F. Sowa (ed.), Principles of Semantic Networks. Morgan Kaufman Publishers, San Mateo, California, 1991. Pp. 45-94.