Introduction to Text Graphs

Esko Nuutila and Seppo Törmä
Helsinki University of Technology, P.O.Box 5400, FIN-02015 HUT, Finland

esko.nuutila@hut.fi, seppo.torma@hut.fi

Abstract

We introduce text graphs, a simple and easy-to-learn notation to represent complex conceptual content. Text graphs have been designed to replace essays and concept maps as constructivistic assignments in our computer science courses. The advantages of text graphs over previous similar notations is accuracy and well-defined meaning, which make them especially suitable for representing technical content.

1 Introduction

Text graph notation provides a way to write text divided up into the nodes of a graph structure. The graph offers an explosion view to the contents of the text. When used properly, it makes explicit the meaningful parts of the text and the relations between these parts.

The *nodes* of a text graph contain fragments of text. These fragments may include so-called *anchors* that are placeholders for other text fragments. The *edges* of a text graph connect nodes and anchors to each other.

Figure 1 below contains an example of a text graph that describes concepts related to the file system of a computer. There are seven nodes and eight edges, and some of the nodes contain anchors shown as small rectangles.

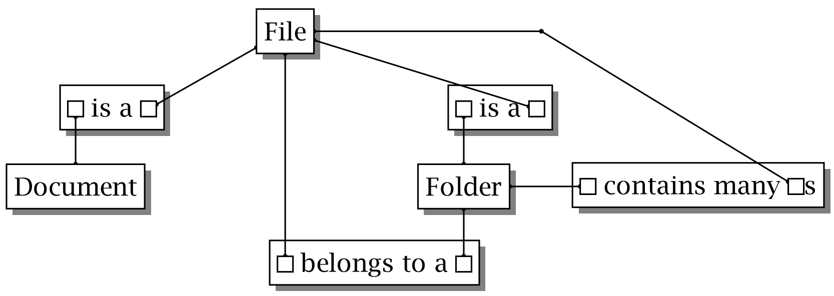


Figure 1: A text graph of file system concepts

It is possible to define a *meaning* of a text graph with respect to *natural language*, namely, the language used in the nodes of the graph. This enables us to specify if the text graph is valid (has any meaning at all) and argue whether it is true or false.

The natural language meaning of a text graph can be derived as follows. Each node that has no edges connected to it (but may have edges connected to its anchors), has a *text expansion* that corresponds to a sentence in natural language. The text expansion of a node can be created by replacing the anchors in the node with the text expansion of the node connected to the anchor. The text graph in Figure 1 corresponds to the following set of sentences:

```
Document is a File.
Folder is a File.
File belongs to a Folder.
Folder contains many Files.
```

Despite their simplicity, text graphs turn out to have many different *applications*. The nodes may contain different kinds of text fragments denoting concepts, individuals, propositions, or questions. This allows the text graphs to be used for various purposes such as concept mapping, argumentation diagrams, or text annotation.

In addition, text graphs suit to different *modes of use*. The correspondence of a text graph with a set of natural language sentences can be used in both directions. On the one hand, an existing text can be analyzed to produce a text graph that reveals its conceptual or propositional structures. On the other hand, a text graph can be produced first and then be used to derive the text that describes the essential structures of the domain.

2 Background and motivation

We have developed the text graph notation for the students of our computer science courses. The goal is to enable the students to externalize the content of their cognition in an easy-to-learn but accurate manner. During the past few years, we have attempted to create new kinds of constructive assignments for introductory programming courses to improve the students' understanding of the central programming concepts (see, for example, Nuutila et al. (2003)). The idea of text graphs arose in autumn 2002 as a result of our earlier experimentations with *mind maps* (Buzan, 1974) and *concept maps* (Novak and Gowin, 1984; Novak, 1998)

Before these experiments we used to give the students assignments to write *essays* about conceptually complex topics faced in the courses (e.g., "Exception handling in Java"). Essays, however, offer only an indirect way to develop conceptual understanding. Students end up spending time in producing fluent and readable text instead of focusing on the refinement of the concepts and their relations. In addition, the teachers waste time in separating the level of the conceptual understanding from verbose rhetoric and surface quality of the text.

The mind maps and concept maps provide a more direct way to work with concepts. However, mind maps — that consist of concepts and unspecified associations between them — soon turned out to be far too vague a notation for working with systems of concepts, especially in a technical domain such as computer science. It seems that mind maps are primarily a technique for taking notes and sometimes a useful memory aid.

Our experiences with concept maps were encouraging but not completely satisfactory. Concept maps are, however, the closest precursors of text graphs: the overall nature and intended uses of these notations are similar. Like text graphs, concept maps are easy-to-learn and are intended for the analysis and refinement of complex conceptual material. They seldom are finished with the first attempt but are revised as the understanding of the domain improves.

Concept maps were developed by Joseph Novak (Novak and Gowin, 1984; Novak, 1998). A concept map is composed of concept nodes connected to each other with named relation nodes. Each relation R between concepts A and B could roughly be read as "A is in R with B". Figure 2 gives an example of a concept map.

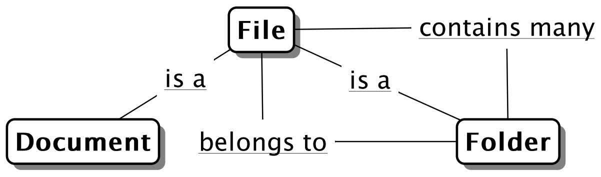


Figure 2: A concept map describing the file system concepts

When we instructed our students in the use of concept maps and studied the maps they produced, we repeatedly encountered the following kinds of problems in this notation:

- *Semantic vagueness.* The relations seldom specify uniquely the intended reading. The readers must use their background knowledge and imagination to read a relationship. As a result, different people — and also the author of the graph at different points of time — may produce different readings of the same relationship.

- *Expressive inadequacy.* Fundamentally, concept maps are meant for representation of concepts and *binary* relations between them. In our experience, the attempt to represent other kinds of relations leads to cumbersome or unnatural maps.

- *Concept/relation separation.* The relations can be used to make propositions about concepts, as in "File is a Folder". However, it is not possible to specify relations that make propositions about other relations in a concept map. This separation between concepts and relations probably makes the use of concept maps easier in simple cases but reduces their applicability to other more advanced uses.

Åhlberg (2002) has presented some improvements to concept mapping. He stresses non-hierarchical nature of knowledge, the importance of accuracy in the labels of concepts and relations, and that each concept should appear only once in a map. One concrete technique is to draw concept nodes inside others that represent more complex concepts. Unfortunately, this is only a partial solution, since a node cannot be inside more than one other node.

Text graphs do not have these problems, as will be shown later on in this paper. Before that, we briefly contrast text graphs with other techniques that have similar outlook or purposes.

Knowledge representation research is an area where different kinds of graph notations for conceptual information has been developed. The goal in knowledge representation is to encode knowledge in a form whose meaning can be processed by a machine. In practice, when a graph notation is used, it should be possible to interpret it as a set of sentences in some logic language to enable the use of automatic inference procedures. Many different graph notations have been developed, with the name of *semantic networks* (Bobrow and Winograd, 1977; Brachman and Schmolze, 1985; Fahlman, 1979) or *conceptual graphs* as specified by John Sowa (Sowa, 1984, 1999). All of these notations are extensive and cannot be characterized as easy-to-learn.

It should be stressed that text graphs have not been intended as a notation for knowledge representation. While knowledge representation aims to transmit to a machine the kind of domain information that it can independently act on, text graphs are meant to aid the human cognitive processing. By constructing a text graph, a person can externalize the content of his or her cognition and in the process develop better understanding of that content. To serve this goal, the simplicity of the notation and a clearly defined meaning to the user are crucial.

Text graphs may also appear to have similarities with *hypertext graphs* that represent associations between (hypertext) documents. However, text graphs do not deal with inter-document relations but with the *internal structure* of documents. The most widely used hypertext system, *The World-Wide Web*, uses a variety of *markup languages* (*HTML* or *XML*) for the internal structuring of the documents. Markup languages are based on an implicit model that a text is an *ordered hierarchy of content objects (OHCO)*, i.e., the structure of a document can be represented as a *tree* whose nodes are *content objects* (for example, paragraphs, figures, or headings) and the children of each node always come in a specific *order*.

Renear et al. (1996) argue that the *OHCO* model, even if refined and extended to cover multiple simultaneous hierarchies, is insufficient to model many important structures in text. Based on our experience it seems evident to us that the *relationships between concepts* or the *chain of reasoning* presented in a text are difficult if not impossible to represent in a hierarchical manner; instead, more general graph structures are required. Indeed, markup languages offer ways to define identifiers for the nodes in the tree and to refer to these identifiers as a way to break out of the hierarchical strait jacket. In text graphs this is possible in a simple and natural way without the overhead of notational tricks required in markup languages.

Recent developments of the *Semantic Web*, that combines knowledge representation techniques with hypertext systems, have some common traits with text graphs. For example, the basic model of the *Resource Description Framework* (Lassila and Swick, 1999) is a graph structure of nodes (called *resources*) connected with edges. Each edge can be interpreted as a proposition composing of the *relation name* and the *subject* and the *object* (or value) of the proposition. Thus, in principle it is possible to produce a natural language sentence that documents the meaning of the proposition. Moreover, the relations can be *reified*, making it possible to make propositions about propositions.

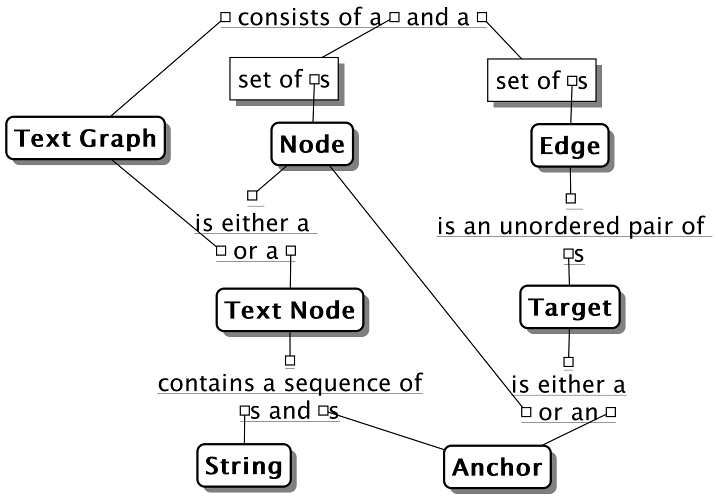
However, the emphasis in the Semantic Web research is in knowledge representation techniques and automatic reasoning. Its goals as well as tools differ from those of text graphs.

3 Structure and Meaning of Text Graphs

Since we have claimed that text graphs are easy to learn, and can be used to accurately present conceptual and propositional content, we decided to use text graphs below to describe the structure, properties, and meaning of text graphs themselves. The additional advantage is that we can present multiple examples of text graphs.

3.1 Basic structures

The basic structures of text graphs are described in Figure 3. The nodes that have different roles are shown in different graphic styles.

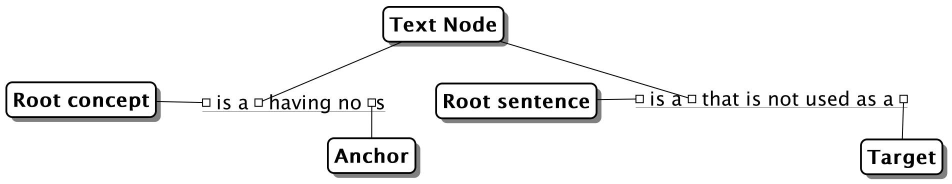


```
Text Graph consists of a set of Nodes and a set of Edges.
Node is either a Text Graph or a Text Node.
Text Node contains a sequence of Strings and Anchors.
Edge is an unordered pair of Targets.
Target is either a Node or an Anchor.
```

Figure 3: The basic structures of text graphs and the resulting text expansion

Note that the target of an edge may be another text graph. This allows the building of nested text graph structures. The nodes in a text graph can be in different roles. Figure 4 identifies two important categories of nodes: *root concepts* and *root sentences*. Root concepts are shown in Figures 3 and 4 inside rounded rectangles and root sentences without a border.

There can be nodes that are neither root concepts nor root sentences, as the “set of” nodes in Figure 3. The non-root nodes can be structural concepts, non-independent sentences, or parts of other sentences. If text graph only contains text nodes without any anchors, they are all both root sentences and root concepts.

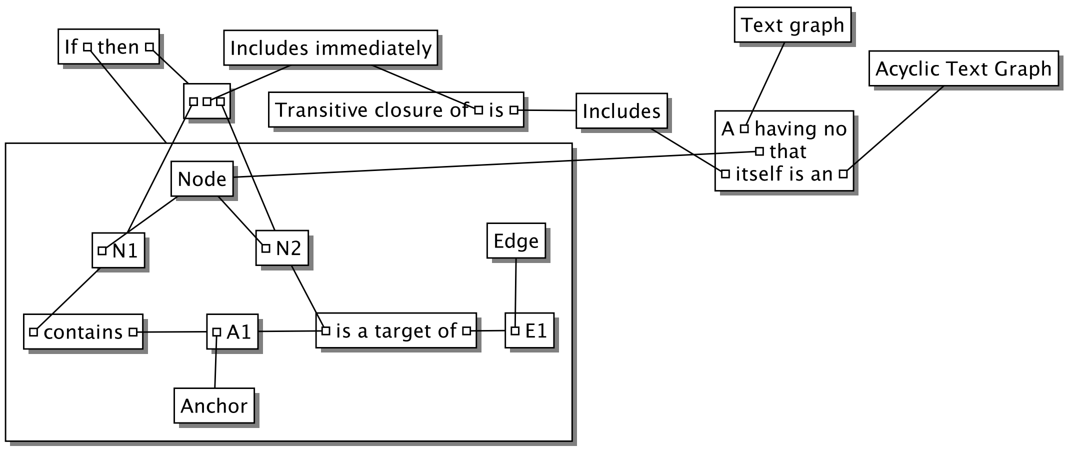


```
Root concept is a Text Node having no Anchors
Root sentences is a Text Node that is not used as a Target
```

Figure 4: Root concepts and root sentences of text graphs

3.2 Meaning through text expansion

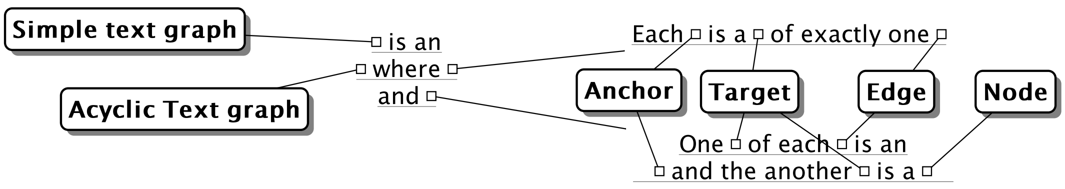
A *simple text graph* is one where each edge connects a node and an anchor, each anchor is the target of exactly one edge, and the transitive inclusion relation contains no cycles, as specified in Figures 5 and 6. Note that the text graph in Figure 5 contains a subgraph. An edge connects the first anchor in the `If`-node to the whole subgraph; there are also edges that connect anchors outside the subgraph to nodes inside the subgraph. This is quite legal by the definition of *Node* in Figure 3.



```
If Node N1 contains Anchor A1 and Node N2 is a target of Edge E1 and Anchor A1
is a target of Edge E1 then Node N1 Includes immediately Node N2.
Transitive closure of Includes immediately is Includes.
A Text graph having no Node that Includes itself is an Acyclic Text Graph.
```

Figure 5: Acyclic text graphs

The meaning of a simple text graph is specified with respect to the (natural) language used in the nodes. We define that the *meaning of a text graph is the meaning of its text*



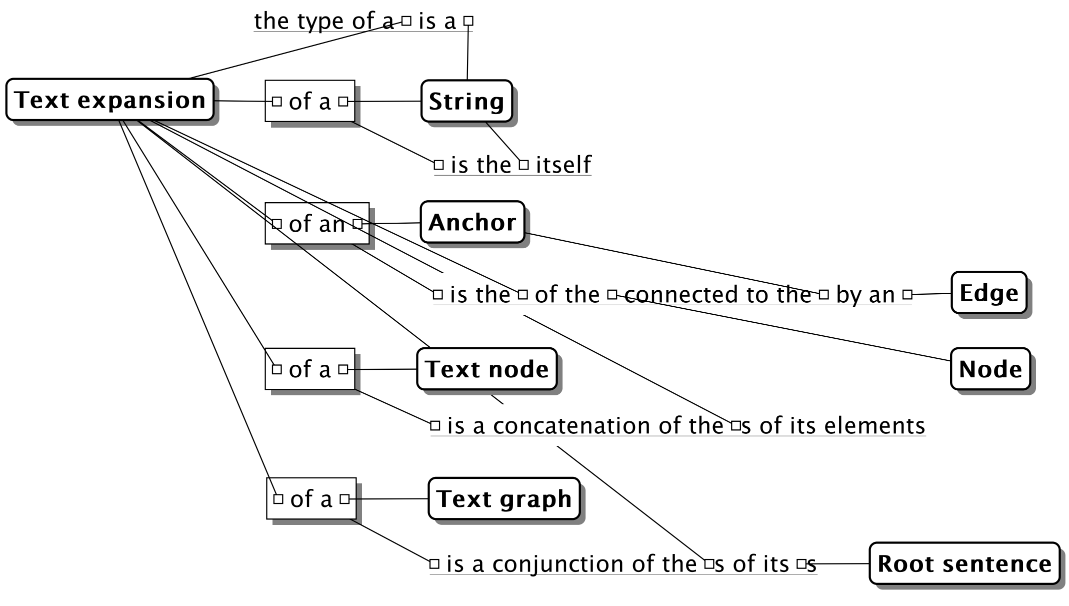
```
Simple Text Graph is an Acyclic Text Graph where each Anchor is Target of
exactly one Edge and One Target of each Edge is an Anchor and the another
Target is a Node
```

Figure 6: Simple text graphs

expansion. The text expansion is a string containing expressions in a natural language and its meaning possibly depends on the interpretation of the people who read it. However, the following can be said: (i) Two different text graphs mean the same for an individual reader, if their text expansions mean the same for that reader. (ii) One text graph means the same for two different readers, if its text expansion means the same for them.

The text expansion of a simple text graph can be derived as specified in Figure 7. These rules assume that there is exactly one edge connected to each anchor and the other end of that edge is connected to a node. The text expansion of a text graph is unique, if the order of the sentences produced is not considered relevant.

Note that the conjunction of text expansions is handled differently in a top-level graph and in the subgraphs. In top-level, the text expansions of the root sentences are ended by full stops and concatenated, whereas in the subgraphs they are connected with the word 'and' (see Figure 5). There are some other linguistic issues in the forming of the text expansion, which are outside the scope of this paper.



```
The type of a Text expansion is a String.  Text expansion of a String is the
String itself.  Text expansions of an Anchor is the Text expansion of the
Node connected to the Anchor by an Edge.  Text expansion of a Text node is the
concatenation of the Text expansions of its elements.  Text expansion of Text
graph is the conjunction of the Text expansions of its Root sentences.
```

Figure 7: A text graph describing the text expansion

A text graph is *valid*, if it has a text expansion and the size of the text expansion is finite. Every simple text graph has a finite text expansion (Figure 7) and is therefore valid.

3.3 Multiply connected anchors

As a shorthand it is possible to draw text graphs where anchors are connected to multiple edges. This kind of graphs can always be automatically converted to simple graphs as shown in 8. Therefore the text expansion rules shown above are sufficient to specifying also the meaning of these non-simple cases, as long as the graphs are acyclic.

Figure 8 shows how a node containing two multiply connected anchors (in the left) is split into a set of nodes (in the right) that is the cross product of the connected nodes.

When one node is transformed into multiple singly connected nodes, it is possible that other nodes will get multiply connected anchors as the result. The transformation must thus be propagated in the text graph but only toward the root sentences. The root sentences may be multiplied but that does not result in any additional multiply connected anchors. Therefore the propagation always terminates.

As a consequence, *every acyclic text graph is valid.*

3.4 An example

Figure 9 shows a text graph that contains the Peano's five postulates. It should be noted that the four central concepts are clearly shown. The use of the concepts in the postulates is clear and it is easy to see that `Natural number` is the central concept. An attempt to represent this content with a concept map would lead to a very unnatural map.

4 Text graph tool

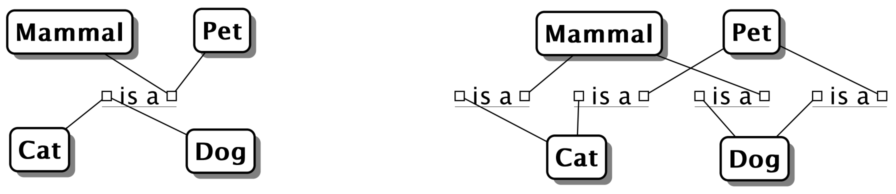
Creating a graphical presentation like a text graph by paper and pencil is a tedious task. It is difficult to determine in advance the space requirements and the relative positions of the different parts of the presentation. This problem is greatly magnified when the understanding of the presented structure evolves over the time. The manual labor and the problems with the layout may take most of the capacity of the author.

To enable the authors to pay most of their attention on the conceptual content of the subject matter, a suitable drawing program is a necessity. There is a freely available drawing program called CMapTool (Novak, 2001) which in our experience is indispensable when working with concept maps. However, it is not completely adequate for text graph editing. Text graph editing requires means to flexibly modify the relative positions of the elements so that the edges remain connected to the nodes and anchors. In addition, it requires support for nested graphs and fluent editing of mixed content of character strings and anchors. CMapTool does not satisfy the latter requirement. Common drawing tools lack both of these capabilities or provide them in a cumbersome manner.

To satisfy these requirements, we have implemented a text graph editor called TGE. The figures in this paper have been drawn with TGE. The basic capabilities of TGE include node creation and deletion, editing of mixed content of character strings and anchors in an Emacs-like manner, creation and deletion of edges, and moving the graph elements maintaining the topological relations.

More advanced functionality of TGE includes:

- Direction constraints for the edge segments and automatic constraint propagation. This allows the creation and maintenance of edges that consist of orthogonal segments.
- Node and edge styles. In the figures in this paper, the core concepts and the core statements have different styles to highlight their different roles.
- Unlimited undo and redo functionality, and complete logging of the editing operations. In addition of helping the editing task, these capabilities make it easier to have different versions of the same text graph.



```
Cat is a Mammal.  Cat is a Pet.  Dog is a Mammal.  Dog is a Pet.
```

Figure 8: An example of a multiply connected anchors and resulting simple text graph

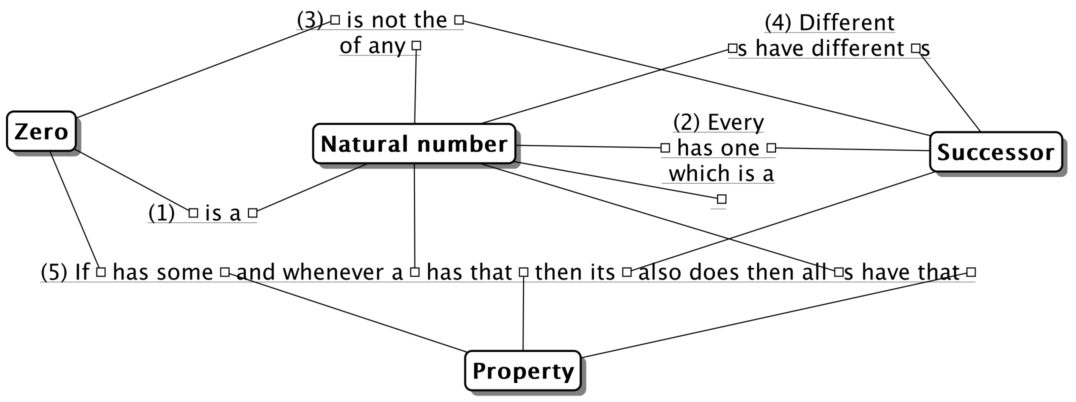


Figure 9: Peano's five postulates

- Multiple simultaneous graphical layouts of the text graph. With this functionality, the user can experiment with different layouts for the same graph, while maintaining the topological structure and the textual content of the graph intact.

TGE is implemented in Java. It stores the text graphs and editing logs into Scheme files, and uses Kawa, a free Java-based Scheme system (Bothner, 2003) for processing the Scheme expressions. Currently we have only a test version of TGE, but a more reliable product version will be freely available later.

5 Summary

We have introduced text graphs, a simple notation to represent complex conceptual content. Their goal is to aid human cognitive processing, for instance, in educational settings. The main advantages of text graphs over previous notations with similar goals are *accuracy* and *well defined meaning*. These properties make text graphs especially suitable for technical domains like computer science. We have also implemented a text graph editor called TGE.

We intend to use text graph notation and TGE tool in our future computer science courses. We anticipate that this tool will also be used in other domains such as argumentation analysis (Fischer, 1988), knowledge management, and initial stages of the development of ontologies.

References

Åhlberg, M., 2002. Concept maps and improved concept mapping. Postscript in the Finnish edition of Novak (1998), in Finnish, see also http://www.metodix.com. 3

Bobrow, D. G., Winograd, T., 1977. An overview of KRL, a knowledge representation language. Cognitive Science 1, 3–46. 3

Bothner, P., 2003. Kawa, the java-based scheme system. Version 1.7, http://www.gnu.org/software/kawa/. 8

Brachman, R. J., Schmolze, J. G., 1985. An overview of the KL-ONE knowledge representation system. Cognitive Science 9 (2), 171–216. 3

Buzan, T., 1974. Use your head. BBC, London. 2

Fahlman, S. E., 1979. NETL: A System for Representing and Using Real-World Knowledge. MIT Press, Cambridge. 3

Fischer, A., 1988. The logic of real arguments. Cambridge University Press, Cambridge. 8

Lassila, O., Swick, R., Feb. 1999. Resource description framework (rdf) model and syntax specification. http://www.w3.org/TR/REC-rdf-syntax. 3

Novak, J. D., 1998. Learning, Creating, and Using Knowledge: Concept Maps As Facilitative Tools in Schools and Corporations. Lawrence Erlbaum Assoc., London. 2, 8

Novak, J. D., 2001. The theory underlying concept maps and how to construct them. Version 2.9.1, http://cmap.coginst.uwf.edu/info/. 7

Novak, J. D., Gowin, D. B., 1984. Learning how to learn. Cambridge University Press, Cambridge. 2

Nuutila, E., Törmä, S., Malmi, L., 2003. PBL in an introductory programming course — how to apply the Seven Steps Method. Submitted for publication. 2

Renear, A., Mylonas, E., Durand, D., 1996. Refining our notion of what text really is: The problem of overlapping hierarchies. In: Ide, N., Hockey, S. (Eds.), Research in Humanities Computing. Oxford University Press, http://www.stg.brown.edu/resources/stg/monographs/ohco.html. 3

Sowa, J. F., 1984. Conceptual Structures: Information Processing in Mind and Machines. Addison-Wesley, Reading, Mass. 3

Sowa, J. F., 1999. Knowledge Representation: Logical, Philosophical, and Computational Foundations. Thomson Learning, Stamford, Connecticut. 3