Formal explorations of knowledge intensive tasks

Jansen, M.G.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Introduction

1.1 Background

Over the years Knowledge Engineering (KE) has identified a number of problem types known as tasks such as classification, design, diagnosis, planning etc (see Puppe [71] for an overview). Each task covers a range of problems which can vary considerably from one another. As an example of a task one can think of diagnosis. Within diagnosis there are several problem variations. For example, performing abduction while assuming that there is only one fault which accounts for a malfunction, is known as single fault diagnosis, whereas the absence of such an assumption is characteristic of multiple fault diagnosis.

There is general consensus [71, 82] about distinguishing two categories of tasks: analytical and synthetic tasks. The first covers problem types such as classification and diagnosis, and involves reasoning about an 'existing system' such as the object to be classified in the case of classification, or the artefact to be diagnosed in diagnosis. In synthetic tasks the 'system' has to be constructed. Hence planning involves the construction of a plan, assignment involves the construction of an assignment, etc.

Conceptualizations of a task are usually given in the form of an ontology of the task. A task ontology provides a specification of the vocabulary of a task, which can be used to formulate problems which belong to that task. For example, a task ontology for classification provides a specification for notions as "class", "observation", "explains", "attribute" etc. These task ontologies are often presented in an informal, or semi-formal way [64] and leave considerable room for interpretation.

With each task one can associate several methods, known as task methods [82], together with typical domain knowledge schemas or conceptualizations. In the commonKADS methodology [82] each task has been given a task template. This template can be seen as a prototypical method for the task which can be taken as a starting point for the development of more specific task methods.

Ideally, when trying to solve a knowledge intensive problem, one first determines to which task it belongs. Next, one inspects the template and adapts it where needed. Finally, one instantiates the method by supplying the various types of knowledge in the proper representational format.

However, such an ideal picture is often cumbersome in practice. Therefore a better understanding of specific tasks is still needed. This thesis focuses on this subject.

Tasks are generally described as consisting of three parts: A specification of the knowledge used, the types of in- and output, and a specification of the goal to be reached.

The task method describes how a goal is realized through a decomposition into substructures, together with some flow of control. These substructures are either tasks themselves (subtasks) or
inferences (primitive elements of which the functionality is in no need of further decomposition). A control structure describes in what order these should be executed.

The default method for a task corresponds roughly to the notion of Problem Solving Method (PSM) in the literature [25, 84, 28, 11]. Examples of PSMs are generate & test, hill climbing and propose & revise. The main difference is that PSMs are usually more general, and as such they are not directly linked to the application of a task. Task methods can therefore best be seen as instantiated PSMs [82]. Hence, a task method describes a general method for problems belonging to the task, using the vocabulary specified in the task ontology. PSMs are usually more generic and can use task-independent vocabulary. In order to instantiate a PSM one has to map the vocabulary of the PSM to the one of the task ontology.

It should be stressed that the identification of tasks, and the description and use of the methods takes place on a high level of abstraction. They are the results of experience of designing and building knowledge based systems, rather than the product of formal, theoretical investigations. In that sense PSMs are similar to design patterns in the field of object oriented programming [35]. Design patterns are presented as program structures which have proved their value in practice and which can often be identified in complex software systems. Therefore it is worthwhile to describe and classify them and make them available for future re-use.

Similarly, the motivation for characterizing and describing PSMs is usually based on pragmatic ideas. They are not finely tuned algorithms but general methods which are expected to be adapted to the peculiarities of a given problem. As such they should speed up the design of complex knowledge based systems.

Formal methods have been used to present analyses of tasks from a "top-down" perspective. In such an approach tasks are usually decomposed and then a specification for the substructures, or inferences, is given together with some 'glue'. An example is the work of Aben [1]. His approach consists of breaking up tasks into a number of inferences first. He then presents a unified, formal framework in which all these inferences are described in detail.

Another example of an approach which aims at presenting a single, unified description of tasks, is the work on the Unified Problem Solving Language (UPML) [26]. This architecture provides one general framework for the description, development and re-use of task methods. In this project one has experimented with formal languages as a specification language for some of the notions used in the architecture. Related to this is the work of Fensel [25] which presents a unified view on PSMs from a top-down perspective.

Such a unified view on tasks is often motivated by the prospect of enhancing the development of knowledge based systems. In contrast to this, one can distinguish a more theoretical approach which is concerned with the formal analysis of individual tasks. In such an approach one analyzes tasks individually and different tasks can be described in different formalisms.

As an example of a more theoretical task-oriented approach we mention the work of ten Teijen [88] on diagnostic problem solving. Here some parameters for diagnostic methods are identified. By varying the instantiations of these parameters one can generate a spectrum of methods for diagnosis.

Also, in the same work, the formalism of approximate reasoning [79] was used, and adapted to describe characteristics of diagnostic methods. This approach differs from the ones above since it focuses on a single task only. This is an example of how an existing formalism can be employed to describe the general reasoning structure of a single task. However, the motivation behind this work consisted of the use of the analysis for the automatic configuration of PSMs in diagnosis.
Another task which features quite heavily as the subject of theoretical research is planning. As an example, we mention the planning-as-satisfiability approach of Kautz & Selman [52] and the relation of planning with linear logic [61]. Such investigations provide insights into the task which are often missing in the pragmatic approach. However, such analyses are seldom linked to the methodologies of KE, and often do not make use of task ontologies and ignore problem variations within the task.

Many of the formal descriptions of tasks within the field of KE have a rather static nature. Often, first or second order predicate logic is used to capture the functionality of the task and inferences. Attempts to provide a dynamic description of knowledge modelling have been made. An example is the ML\textsuperscript{2} language [92] which provided a formal description of the KADS knowledge model. Dynamic logic was used here to describe the dynamics of the knowledge models. However, in this language it is difficult to give a clear and intuitive account of the various stages of the problem solving process.

In this thesis we explore in what ways existing formalisms can be used to analyse tasks within the framework developed by the pragmatic approach. By this we mean that we will make use of ontologies and are aware of problem variations within the task. We would like to characterize the task in knowledge-level [67] terms. That is, we give a description of which knowledge is present before, during and after solving a problem which belongs to a certain task. Hence, we will present a static as well as a dynamic account of knowledge-level problem solving. The aim is not to develop new, or faster Problem Solving Methods but to acquire a better insight into the nature of tasks.

1.2 The Problem

In general terms the problem statement of the thesis can be formulated as follows:

**HOW CAN WE GIVE A KNOWLEDGE-LEVEL CHARACTERIZATION OF THE PROPERTIES OF KNOWLEDGE INTENSIVE TASKS?**

The aim is not to develop a new logical language or formalism, but to explore how existing formalisms and approaches can be used to acquire new insights regarding knowledge level task specification.

The problem statement can be refined by introducing three related questions:

1. **WHAT DOES A CONCEPTUALIZATION OF A TASK LOOK LIKE?**
   By a "conceptualization of a task" we mean a characterization of the notions involved in problem description. The conceptualization of a task should include a task-ontology in which the task-specific vocabulary should be specified. In addition it should include some (not necessarily all) criteria for a solution to some typical problems which belong to the task. Finally, the conceptualization of a task may contain a task-template, or some general description of what problem solving for that task would look like. Note that a conceptualization of the task is often done on a high level of abstraction and is very much a knowledge-level construction.

2. **HOW DO WE REPRESENT THESE KNOWLEDGE-LEVEL CONCEPTUALIZATIONS IN A FORMAL WAY?**
The next step is to choose an adequate formal representation of these conceptualizations in order to present a thorough characterization of a task. Such a formalism forces one to be precise about the knowledge-level constructs. In addition we would like that the properties of the formalism would tell us something about the nature of the task, and that proof methods for the formalism have some relation with problem solving of the task.

3. **How can we use these representations to acquire a better understanding of the task?**

As formal specifications of knowledge-level conceptualizations force one to be more precise, one may discover that new distinctions can be made and new variations can be formulated. For example, given some initial criteria for solutions a formal analysis may lead to a more rigorous and systematic distinction between various criteria. This can lead to the systematic generation of a spectrum of the problems which are covered by the task description.

Another point of attention is that a formalism can have several proof methods associated with it. It is interesting to see whether one of these could be used, or adapted to describe task methods.

### 1.3 Approach

In order to answer the questions above we will focus on a limited selection of specific tasks. Some tasks like diagnosis [8, 21] and planning [4] have been extensively described and analyzed in the literature. We therefore focus on tasks which have been less extensively explored, like classification and assignment. We have taken care to include both an analytical task (classification) and a synthetic one (assignment). The results of the explorations of these tasks will be generalized to some extent in order to answer the questions in the previous section.

We want to stress that we want to give a knowledge-level description [67] of the task in a dynamic, formal fashion. Another point to note is the way we use logic, and formal methods in general. Our approach is aimed at the understanding of a knowledge-level specification of tasks, and formal methods are used as means to that end, not as an end in themselves. Hence, we will be quite pragmatic in the use of existing formalisms.

The steps which we will take to arrive at the answers to the research questions are the following:

- **Conduct an informal study of a knowledge-intensive task at the knowledge level.**
  As an introductory step, a task is selected and analyzed in detail, using the traditional semi-formal specification methods used in knowledge engineering. Special attention should be paid in this exercise to the nature of the ontologies and Problem Solving Methods involved. This should give adequate input to the needs for logical formalizations.

- **Show how an adequate formalism for a task is chosen.**
  The result of the previous step will serve as input for the choice of an adequate logical representation of task-specific reasoning.

- **Provide a specification of the range of problems within a task.**
  We will provide a spectrum of problems for two tasks, one *analytic*, the other a *synthetic*
task. This should provide a specification of how problems can vary within a task and how they can be elegantly described, classified and compared. This description of problem variations should be done in terms of the elements of the task ontology.

- **Provide a more general framework for the dynamic description of task-specific reasoning.**
  A more dynamic approach to the description of knowledge intensive tasks will be presented. Problem solving will be described in terms of increasing knowledge about possible solutions. We will use the notions of 'knowledge state' and 'state transition' in these descriptions.

### 1.4 Contents of the thesis

The thesis will be divided into nine chapters. Chapters 2, 4, 5 are devoted to analytical tasks, whereas 7 and 8 are about synthetic tasks. Chapter 6 contains the description of a framework aimed at a dynamic description of knowledge intensive tasks based on the idea of information flow.

Finally, chapter 9 contains the conclusions and answers to the research questions formulated in the current chapter.

**Chapter 2 - Rocky III**  
This chapter describes the results of research performed as part of the Sisyphus III project [46]. Researchers in the knowledge engineering community were asked to develop a classification system for igneous rocks. Comparing the various approaches taken by different researchers was the research goal of the project. This chapter presents a contribution to the Sisyphus project. As part of the thesis the chapter provides an introduction into the use of traditional specifications of ontologies and problem solving methods. More specifically the task ontology and classification methods described here, can serve as input for chapter 3.

**Chapter 3 - Adopting tableaux for classification**  
In the chapter 2 a classification problem was looked at in detail. In this chapter we focus at the classification task itself and give a more general characterization. It is shown how classification problems can be represented in such a way that they can be solved with (manipulated) analytic tableaux. Two problem solving methods for classification are discussed: weak and strong classification. For both methods procedures in terms of tableaux are presented. The similarity between strong classification and abduction is highlighted.

**Chapter 4 - A spectrum of classification methods**  
In the previous chapters several variations of classification were described. This leads to the question what other problem variations are possible. This chapter describes the construction of a spectrum of classification criteria, built up from elementary building blocks which are part of the ontology of the task.

**Chapter 5 - Using strict implication in background theories for abductive tasks**  
Domain knowledge in abductive tasks is often represented in classical logic, using the material implication symbol ‘→’ in a specific way. We explore the objections against the use of material implication (first raised by Lewis in the 1930’s [56]) and see if these affect the use
and intended meaning of the representation. As one of the possible alternatives we look at abduction with strict implication.

**Chapter 6 - A dynamic approach to specifying analytic tasks**  
In this chapter we provide a more general analysis of tasks in terms of knowledge about the solution. We show how problems within a task can be given a knowledge level characterization in a dynamic way. Problem solving is described as changing knowledge on the basis of new information and pro-active behaviour.

**Chapter 7 - Towards an ontology for knowledge intensive assignment problems**  
In this chapter we will provide a task ontology for assignment problems. The main emphasis is on the characterization of different forms of constraints and preferences. The ontology covers both assignment problems from the field of Constraint Satisfaction, and the family of Stable Marriage problems [44].

**Chapter 8 - Formalizing group assignments**  
We look at a subproblem of assignment known as grouping. This strategy is particularly interesting in combination with the use of abstractions in the domain knowledge. We explain in detail how abstractions and groupings can be treated as partitions of a set.

**Chapter 9 - Conclusions**  
This chapter contains the conclusion of the thesis and answers the research question formulated above.