Beyond the Knowledge Level: Descriptions of Rational Behavior for Sharing and Reuse

Franz Schmalhofer[1], J. Stuart Aitken[1] and Lyle E. Bourne Jr.[2]

[1] German Research Center for Artificial Intelligence, P.O. Box 2080, D 67608 Kaiserslautern, Germany email: schmalho,aitken@dfki.uni-kl.de
[2] Department of Psychology, University of Colorado, Campus Box 430, Boulder, Colo. 80309 email: lbourne@clipr.colorado.edu

Abstract. The currently dominant approach to the sharing and reuse of knowledge strives to develop ontologies, with clearly constrained interpretations. The idea of ontological commitments is based on the knowledge level perspective. Several shortcomings of the knowledge level have been identified (Clancey, 1991). Pursuing Clancey's argument, if KBS are to be situated in ever changing environments, their purposes and significance will change over time and they have to be redescribed accordingly. The behavior descriptions proposed in this paper emphasize coherent and consistent descriptions in some given context, rather than predicting performance from knowledge and goals. When systems are embedded into larger contexts, their behavior is redescribed so that the additional significance is shown. Behavior level descriptions thus provide the flexibility for conceptual changes in a knowledge-based system. We demonstrate how behavior descriptions can be used for documenting KBS and present an example of the documentation of a KBS for elevator configuration.

1 Introduction

It has recently been recognized that the development of large knowledge-based systems which can be successfully used for real world applications requires the sharing and reuse of knowledge. In the so called knowledge sharing effort (Neches *et. al.*, 1991)(Swartout, Neches and Patil, 1993) a vision and corresponding technologies are being developed so that a new system need (and must) not be constructed from scratch but can instead be developed by assembling reusable components of established knowledge such as ontologies that are already embodied in existing knowledge-based systems (Guha and Lenat, 1990; Breuker and Wielinga, 1989). On the basis of shared ontologies, even specific system implementations could be reused when building a new system.

One major working group within the knowledge sharing effort (i.e. the Shared, Reusable Knowledge Bases group) is concerned with developing a language by which a consensus on vocabulary and semantic interpretations of domain models can be established. Gruber (1993) has termed the specification of a vocabulary for a shared domain consisting of definitions of objects, concepts, classes and relations an ontology. Ideally, such shared definitions should be specified at the

knowledge level (Newell, 1982), independent of specific representation languages. On the basis of these assumptions an Ontolingua has been developed. Ontolingua is an implemented system (Gruber, 1993, p. 12-2) for translating ontologies from a declarative, predicate-calculus language into a variety of representation systems like a KL-ONE type system

Sharing and reusing knowledge which is specified in some common logic that is independent of any specific representation language is certainly of pivotal importance for developing successful knowledge-based systems. However, Clancey has recently pointed out a number of deficiencies of current knowledge level specifications and proposed respective re-interpretations (1989). Prior to this criticism, Bourne has already argued that the postulate of pure symbol manipulation processes does not set the right stage for a better understanding of complex systems. He furthermore proposed behavior descriptions which would yield a better understanding of the human conceptual behavior (Bourne, 1969; Bourne, Ekstrand and Dominoski, 1971; chapter 1) and which are better suited for building large knowledge-based systems.

Although, proponents of the information or symbol processing hypothesis promptly refuted the notion of behavior descriptions (Newell, 1969), recent research clearly indicates the value and the promises of behavior descriptions and the underlying scientific idiom. Maes (1993) has recently compared knowledge-based to behavior-based artificial intelligence. Behavior-based systems excelled in that such systems can be understood as a part of the environment. Their performance is consequently emerging from the interaction of the system and its environment. They are autonomous open systems and their symbols are grounded in the environment.

The goal of this paper is to provide a formalization of behavior descriptions and to show how the sharing and reuse of knowledge can be accomplished in a ever changing environment. We will first summarize Newell's knowledge level conceptualizations and point out a number of shortcomings. Whereas knowledge level descriptions, which are often viewed as function-structure blueprints, describe an abstract mechanism, behavior descriptions specify the significant relations among various parameters of behavior. After a general exposition of behavior descriptions, it is exemplified how behavior descriptions can be used for the documentation of artificial systems (expert systems and computer programs in general). The sharing and reuse of knowledge is then discussed when the functionality of a knowledge base is to be extended and when a knowledge base is used for purposes, which were not considered at design time.

2 Newell's knowledge level

In his influential paper, Newell (1982) has introduced a level of computer system description called the "knowledge level". Since this time, describing artificial and human systems as a knowledge system has become an important goal in expert system research (Clancey, 1985; Breuker and Wielinga, 1989) as well as in cognitive psychology (Pylyshyn, 1984; Anderson, 1990). When establishing a

knowledge level description, a natural or artificial system "is viewed as having a body of knowledge and a set of goals, so that it takes actions in the environment that its knowledge indicates will attain its goals"(Newell, 1992, p. 426). Knowledge systems are one level in the hierarchy of systems that make up an intelligent agent. Lower level descriptions such as the symbol level specify how a knowledge level system is realized in mechanisms (i.e. information processing and representation). The symbol level is described by a memory, symbols, operations, interpretation processes and perceptual and motor interfaces (Newell, 1982). Through knowledge level descriptions, Newell has thus provided us with a possibility for uniformly characterizing natural and artificial systems.

The key assumption underlying the knowledge level is the notion of an idealized rational agent. A rational agent is assumed to have the following attributes: 1) The agent has the ability to perform a set of actions in some environment. 2) The agent has goals about how its environment should be. 3) The agent has a body of knowledge. Its body of knowledge is about the environment, its goals, its actions and the relations between them. The agent also knows all those facts that are a deductive consequence of its body of knowledge 4) The principle of rationality is its single law of behavior. It describes which actions the agent will perform: "If an agent has knowledge that one of its actions will lead to one of its goals, then the agent will select that action" (Newell, 1982, p.102).

The behavior of an agent is a sequence of actions taken in the environment over time. By applying the principle of rationality, one can presumably predict the future behavior of an agent from its knowledge and its goals. Consider for example a chess player as such an ideal rational agent. It has the ability to perform a set of moves on the chess board. It has the goal of winning the game. Its body of knowledge consists of the rules of the chess game, i.e. the starting positions and the legal moves and everything that is deductively derivable from this body of knowledge. Therefore, for every chess board constellation, it knows which move will make it win the game. With the principle of rationality, it is thus predicted that it will play the perfect game of chess.

The assumption of such idealized rational agents has been shown to produce (at least) two substantial problems: 1) Important knowledge differences cannot be expressed in a knowledge level descriptions. The symbol level must consequently be used to denote these important distinctions. This is the problem of confounded knowledge differences. 2) A faithful implementation of a knowledge level description at the symbol level would often require unlimited computational resources. The assumptions made about rational agents may thus be too unrealistic for being particularly useful. This applies for describing human cognition (Anderson, 1990) as well as for specifying and implementing knowledge-based systems (Sticklen, 1989). This is the problem of a too unrealistic idealisation. Sticklen has pointed out the problems of the inability to represent control, the potential computational inadequacy and the non-operational characterisation[3].

[3] Schreiber, Akkermans and Wielinga (1990) have rejected these criticisms but we believe the case is not yet closed. We propose skills to avoid introducing control into knowledge descriptions

Confounded knowledge differences: Chess grand masters have spent years of learning for improving their chess game. Unlike beginners, who merely know the rules of the game, chess grand masters are known to have a large vocabulary of different chess board configurations (Chase and Simon, 1973). The chess grand masters thus certainly have more chess knowledge than the beginners. However, in a knowledge level description a la Newell there would be no difference between the beginner and the chess grand master. Because the beginner s body of knowledge would also include everything that is deductively derivable from the rules of the game, he would be said to know everything a grand master can possibly know about chess. In a knowledge level description, a beginner and a grand master would thus be said to have the same chess knowledge.

According to Newell (1982, 1992), the symbol level is described with the following attributes: 1) a memory, with independently modifiable structures that contain symbols; 2) symbols (patterns in the structures), providing the distal access to other structures; 3) operations, taking symbol structures as input and producing symbol structures as output; 4) interpretations processes, taking symbol structures as input and executing operations (the structures thereby representing these operations) and 5) Perceptual and motor interface to interact with an external environment. A symbol level description of a chess player includes its memory for the different types of chess constellations. At the symbol level the chess grand master would thus have dramatically more chess patterns in the symbol structures (chunks) than the beginner.

Unrealistic idealisation: In order to substantiate the body of knowledge of a chess player at the symbol level, one must compute everything that is deductively derivable from the rules of the game. This problem is computationally intractable. In other words, a computational device would be needed that has infinite memory and infinite processing resources. Neither humans nor computer systems can be realistically viewed as such computational devices. Knowledge level descriptions are therefore unrealistic idealisations and not very useful. A knowledge engineer, who wants to implement an expert system from a knowledge level description of the ideal chess player, may not even achieve a rude approximation of the program specification given at the knowledge level (Sticklen, 1989). Since resource limitations are a fundamental characteristic of human cognition (Miller, 1956; Norman and Bobrow, 1975; Simon, 1974), a psychologist, who describes human behavior at the knowledge level would most frequently derive predictions, for which no supporting empirical evidence can be found.

The problems with knowledge level descriptions have been known for some time and various solutions have been proposed (Dietterich, 1986; Schreiber, Akkermans and Wielinga, 1990). These solutions refine Newell's knowledge level notion by imposing more detailed structures on an agent's body of knowledge. They do however not address the root of the problem. Although Newell's proposal of describing natural and artificial systems at a uniform abstract level is extremely important for cognitive psychology and artificial intelligence, his formulation of the knowledge level hypothesis is incomplete and/or misdirected. There are four major misconceptions: 1) Knowledge and goals are in themselves

inadequate to fully characterize intelligent systems. 2) Knowledge level descriptions are developed as if intelligent systems were causal systems. 3) For this level of abstract description, the distinction between an agent and its environment is artificial. 4) The knowledge level does not lie directly above the symbol level and there is no tight connection between them. Therefore knowledge level descriptions cannot be reduced to the symbol level (Clancey, 1991).

3 Behavior descriptions

If existing knowledge level descriptions of systems are inadequate, what then does it take to give an adequate and complete description of intelligent behavior? Knowledge is important of course; an intelligent system knows the important facts of its operative domain. But knowledge alone is inert; it does not act on its own. There are other equally important parameters of intelligent behavior, which include, at a minimum, skill, intention , (goals or purpose) and performance. To talk about the behavior of any agent, natural or artificial, one is obliged to make, either explicitly or implicitly, some commitment to the knowledge, skill, intention (or goals) and performance of that agent. It is not enough to say that the agent has the pertinent knowledge and acts rationally. Such a description is at best incomplete, and at worst wrong.

3.1 Extending knowledge level descriptions

What do knowledge, intention (or goals), skill and performance entail and what does it mean to call these parameters of behavior? Basically we agree with Newell about the nature of knowledge. The knowledge is the complete set of discriminations, facts, or concepts available to an agent which have been acquired from past experiences. Knowledge implies that the agent can make distinctions between and among objects, processes, events and states of affairs. It has a basis for treating some things in one way, others in a different way.

But, as the foregoing discussion might suggest, knowledge does not exist independently of, or in isolation from the way it is used. Thus, with respect to any discrimination, fact or concept there is a corresponding skill, which represents its use. It might be helpful to think of knowledge then, in Ryle's sense, as "knowing that x is the case". Skill, then, is "knowing how to act on that knowledge" (Ryle, 1949). The difference captured in this distinction is the difference between "knowing that" and "knowing how". As Ryle (1949, p32) has clearly stated knowing how cannot be defined in terms of knowing that.

Similarly, intentions or goals do not exist independently of knowledge and skill. An intention is the want, desire or need to act upon some existing knowledge in a particular way.

The combinations of knowledge, skill and intention do not cause any particular behavior to be what it is. Neither do they in any intelligible way cause an action. Rather these are merely parameters of a behavior description. But to

use the term "merely" is not to minimize their importance. There might be no defensible way to give a valid causal description of behavior.

But we are not yet finished. There is a missing ingredient. To provide a complete behavior description, it is required that some commitment be made to the knowledge, the skill, and the intention of the agent. Knowledge, skill and intentions alone provide only a description of behavior potential. What is required to complete the description of an actual behavior is some performance or action by the agent. Please note, classical behaviorism to the contrary notwithstanding, performance is not equivalent to behavior in this descriptive system. It is merely a component or parameter of behavior, meeting the criteria of consistency and coherence. Rather than causing performance to be what it is, knowledge, skill and intention enable a certain performance in the sense that they make it feasible and intelligible when it occurs. The way we use these concepts entails that each one of them is involved in providing a behavior description.

$$\mathbf{B=R(G,K,S,P)}$$

where B is behavior, G are the goals or the intention, K is knowledge, S is skill, P is performance and R specifies the various relationships among G,K,S and P.

These components do not exist independently, in separate systems or in isolation. The knowledge, the skill, and the intention to engage in any action, x, are tightly interconnected. For every quantum of knowledge that allows for discrimination between x and not-x, there is some way of acting on that discrimination and some possibility that the actor will want to take that action. Thus, it does not make sense to look for independent traces of knowledge, skill and intention in separate brain locations of storage mechanisms.

Consider that most adults are able to recognize a case of piano playing when they experience it, even if they have no musical training. I have sufficient knowledge to distinguish piano from non-piano playing in most circumstances (although there might be cases in which it would be difficult for anyone to tell). Furthermore, I have ways of acting on that knowledge, such as calling it a case of piano playing, or a case of bad piano playing, or simply leaving the room where it is being done. I might have the goal of either approaching, because it is good and I want to hear better, or avoiding, because it offends my sensibilities. If that goal is stronger than all other immediate goals, I will take the indicated action. Having done this, I have completed a behavior episode and anyone (including myself) who observed my action and had reason to know or to think about my personal knowledge, skills and intentions, would be correct in describing what happened as an act of behavior.

Because I know what qualifies as piano playing, I can consider hypothetical cases of piano playing. The fact that I know that I know distinguishes me, as a human being capable of intelligent, intentional action, from an automaton. One might realistically say that the thermostat "knows" how to control room temperature, by turning a switch off and on at appropriate times. But it make no sense whatsoever to say that the thermostat "knows that it knows."

The concepts of knowledge, skill, intention and performance are necessary but possibly not sufficient to give a complete description of behavior. There are other constraints of the system that might have to be taken into account in the general case. For almost all purposes of the present discussion, the basic parameters will suffice.

It is tempting to liken the distinction between knowledge and skill in this descriptive system to the concepts of declarative and procedural memory in the contemporary literature of artificial intelligence (Winograd, 1976) and cognitive psychology (Anderson, 1983; Squire and Slater, 1975; Tulving, 1975). It is common to invoke Ryle to argue that memories can be either procedural or declarative.) Declarative memories are memories for facts, either at a general, conceptual or semantic level or as episodes, embedded in some time/space co-ordinates. Most adult human beings know, for example, the concept of a newspaper, or a meal or justice. Further they can be expected to know (at least some of) what they read in the newspaper about the U. S. presidential election at home last night. In the first case, the knowledge is general, semantic and de-contextualized; in the second case, it is embedded in an episode. In either case, the memory is fact-based, and its recollection is always conscious and deliberately achieved (note that a skill is required for this). Procedural memories are memories about doing something. These are the kinds of memories that support acquired skilful performance, as in piano playing, or repetition priming effects, of the sort that make it somehow easier to process a stimulus the second time around, or some classical conditioning phenomena. The basis for these memory effects is nonconscious. It does not appear to be necessary for one to know or to be able to say much about the skill of piano playing in order to play the piano (of course, one would surely know that he is playing the piano, because that's a fact). Moreover, one need not know about or be momentarily aware of a prior episode with a stimulus in order to exhibit (but possibly not experience) a priming effect. To make the distinction between declarative and procedural memories completely clear, some theorists argue that they belong to separate memory systems and never mix. In some cases, the argument is made that procedural memories derive from or are some compiled version of fact-based memories.

We believe that the declarative/procedural distinction is important, but probably muddled in current theorizing. First of all, while it is correct to cite Ryle in support of this distinction, it should be noted that he had a somewhat different meaning in mind. As in the present system, facts and skills were not independent entities for Ryle. Facts and skills go together. For everything you know, there corresponds a way to act upon it. One does not know a newspaper without some related actions (which can, but need not include reading it). There is no evidence of declarative memory without associated action; further there is no case of intelligent action without a basis in knowledge. Which is not to say that skills and repetition effects do not exist. It is indeed possible to exhibit skill without a lot of useful information you can communicate about it verbally or consciously. Repetition priming effects are quite reliable. But the fact that you appear to be unable to report an earlier occurrence of the pertinent stimulus

rules out only one possible thing you might know and uses only one possible way to assess what you know about that stimulus. There simply is no compelling evidence at the present time that different memory systems contain declarative and procedural memories. Thus, the distinction will require further sharpening of both a theoretical and an empirical sort.

Clancey (1991, p. 386) has already pointed out that knowledge level descriptions should not be identified with causal mechanisms because they are observer-relative. The regularities in the performance of humans or artificial systems should be viewed as characterizations that are descriptive. In agreement with Clancey's arguments, the relationship among the parameters of behavior is not causal. That is, the conjunction of knowledge and intention (or goals) does not cause an action. Rather the relationships that governs this descriptive system are consistency and coherence. As we will see, the component parameters must make sense together for the behavior described to be accepted as rational. Component inconsistency or incoherence produces descriptions that are irrational or incorrect or unrecognizable as intelligent behavior. In a more technical notion, a description of rational behavior thus consists of the unification of the parameters of behavior.

The environment is as the agent perceives it. That is, within this system, there is no point to the assertion that the environment is one thing and the organism is another. There is no point in separating internal from external. Organisms do not just function within environments, they are part and parcel of the environment. Among other things, it is for that reason that the environment is not the same for all organisms. Now of course the environment is not entirely different for different organisms. There are regularities, and these regularities are what gives us some scientific purchase on behavior.

The environment is an interpreted framework within which behavior takes place. The environment exists for an agent only because the agent has some knowledge about what it perceives, some skills and goals to act on that knowledge, and an ability to carry forward with action or movement. Environments differ among people, because people differ in which way they know about what they perceive. But because to some degree knowledge is shared, environments are shared. To the extent that two people have the same knowledge, skills, goals and performance abilities they will perceive the environment in the same way and they will behave in the same way. The implication for knowledge-based systems is that we must build environments into them. The knowledge-based system must contain not only knowledge in some internal sense (and skills, goals and performance abilities), but also knowledge of the environment (something like cognition-environment relationships in standard terminology) in which the system operates.

The notion of knowledge has traditionally been used to mediate between the internal states of an agent and the states of the world. But there need not be any tight coupling between the symbol processing of an artificial system and its behavior description. Depending upon the scope and context of a behavior description an identical symbol processing and performance may result in different

behavior descriptions. And conversely, different symbol processing procedures may be given an identical behavior description.

It is often said that science seeks explanations for natural phenomena. The study of intelligent systems (of the empirical and engineering sciences) is, as it should be, fully as scientific as any of its kindred enterprises. Yet, the system we have presented is called a descriptive system; its goal is to supply complete behavior descriptions. We do not intend, by the use of this term, to diminish its value as a scientific tool. Rather, we are merely following the requirements of good logic. Logically, it is impossible to explain something you cannot in the first place describe to some degree of accuracy and completeness. Because existing systems of description in psychology and artificial intelligence are defective, we need to establish a descriptive idiom that will work. Description must come before explanation. But there is also the possibility that, once an adequate, detailed an complete description of behavior has been achieved, there might be nothing left to explain.

It should be noted that behavior descriptions are not identical to performance. Behavior descriptions encompass the goal, knowledge, skill and performance parameters. Thus, performance is only one of the parameters of behavior descriptions.

3.2 Formal behavior descriptions

In this section we describe the formalization of behavior descriptions. We then show how these descriptions can be used to document computational systems. Documentation is essential for understanding the operation of programs and for their maintenance. Good documentation can aid the reuse of program designs. It can also aid the growth or expansion of the system, and the reuse of program design when the purpose, or use-in-practice, of the system changes. The problem of reusing programs and their designs cannot be solved from symbol level considerations alone. We must relate symbol level computations to abstract level descriptions.

While we are concerned with KBS in complex domains we shall begin by discussing examples in the field of set theory. This choice was made in order to illustrate our ideas clearly, in a concrete and widely understood domain. In section 4 we address the more complex problem of documenting a KBS.

A behavior description is defined by the 4-tuple $< G, K, S, P >$. One or more behavior descriptions may be associated with an agent. The components of the behavior description conform to the following specification, where each description is specified with respect to a particular conceptual theory.

$< G, K, S, P >$ is a behavior description, in a conceptual theory C, where:

G :defines the goal which the behavior description can satisfy,
the formal language is predicate calculus (extended as defined below).
K :defines knowledge related to the solution of the goal
the formal language is predicate calculus.

S :describes skills which perform the computation defined in K, the notation is that of functional programming.
P :describes the performance of the skills, i.e. one or more concrete examples of the input-output relation, the formal description is specified by the programming language.

The predicate calculus is extended by the addition of predicates which represent concepts in set theory, for example, set union is formalized by: $\cup(A, B, C)$. However, for clarity, we shall use the standard infix notation of set theory and write $C = A \cup B$ to represent union. In other domains, predicate calculus may not be the most appropriate formal language. In such cases K and G may be described in any suitable formal language which provides appropriate semantics and rules of inference.

It is required that a function, h', must actually exist at the concrete level, in some programming language, before we can confirm the existence of the skill, h. This guarantees that we can obtain examples of actual performances P of the skills. Performances are indexed by the skill they are produced by, the concrete level function name, and the programming language the concrete function is specified in. This method of documentation includes information which can be seen as validating the concrete level code.

The relationships between the various components of the behavior description are illustrated by example in the following documentations of programs.

3.3 Example 1: Set union.

In the framework of simple set theory the union of two sets, A, B, is a set which contains all members of A and all members of B. Elements which occur in both A and B occur only once in the union set. The computation of the union of two sets can be considered to be composed of two operations: the construction of a set C, obtained from B by removing all elements which also occur in A, and then the combination of A and C to yield the output set D. The former computation we call diff and the latter concatenate. The computation is illustrated diagrammatically in figure 1[4], which shows the relationship between data classes and skills. The documentation of this computation in terms of behavior descriptions is given in table 1.

The goal is a summary of the computation, namely that for any sets A and B the union, D, can be calculated. The knowledge component specifies the distinctions which decompose the goal into an equivalent set of formulae, from which the skills can be derived. Formula i. defines the union operator in terms of set membership ϵ and ii. defines an equivalent breakdown of the rhs. of i. which introduces the set C i.e. the set whose members are in B but not in A. The skill C = diff(B,A) is associated with the distinctions defined by the expression $(f \epsilon C \leftrightarrow f \epsilon B \wedge f \notin A)$. The skill D = concatenate(A,C) is associated with $(f \epsilon D \leftrightarrow f \epsilon A \vee f \epsilon C))$, however in this case the context is important as A and C are guaranteed to be disjoint. This association is behavioral, i.e. the skill effects

[4] This diagram illustrates the method but not the details of the computation.

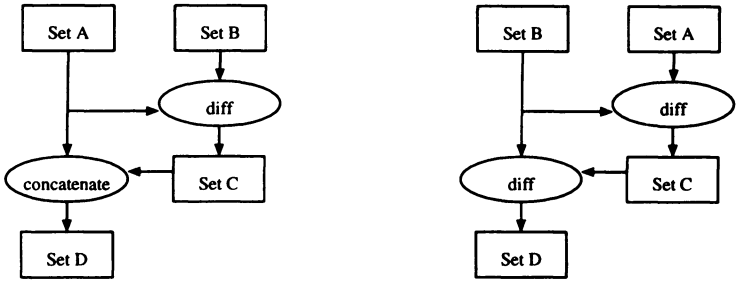

Fig. 1. The Union computation (left) and the Intersection computation (right)

the calculation specified by the knowledge component, as is witnessed by the performance. Where it is possible to prove equivalence between the knowledge component and the concrete function, this is desirable. However, we acknowledge that this may not always be possible in all domains.

LISP, PROLOG and C definitions of the diff and concatenate skills were defined according the decomposition defined above. It is notable that the structure of the conceptual model of the computation was maintained, for imperative, functional and logic-based programming languages.

		Union (simple set theory)
Goal:		Construct $D : (\forall A)(\forall B)(D = A \cup B)$
Knowledge:	i.	$(\forall A)(\forall B)(\forall D)(A \cup B = D) \leftrightarrow (\forall e)(e \epsilon A \vee e \epsilon B \leftrightarrow e \epsilon D)$
	ii.	$(\forall A)(\forall B)(\forall C)(\forall D)$ $((\forall e)(e \epsilon A \vee e \epsilon B \leftrightarrow e \epsilon D)) \leftrightarrow$ $((\forall f)(f \epsilon C \leftrightarrow f \epsilon B \wedge f \not\epsilon A) \wedge$ $(f \epsilon D \leftrightarrow f \epsilon A \vee f \epsilon C))$
Skill:	a.	C = diff(B,A), D = concatenate(A,C)
Performance:	1.	A = '(1 2 3 4) B = '(4 3 5 6) D = '(1 2 3 4 5 6) [skill = a, set-union, LISP]
	2.	A = '(a b c) B = '(d e) D = '(a b c d e) [skill = a, set-union, LISP]
	...	

Table 1. The behavior description for Union.

It is also possible to define set union as one skill C = union-1(A,B), that is, the skill corresponds to the rhs. of i. In this case the function, at the concrete

level, is a synthesis of the diff and concatenate functions. The performance of this new concrete level function is not necessarily the same as that defined in 1. In fact, in the case of LISP, the performance differs in the ordering of the elements of the output list, D. This difference is not significant at the abstract level, in the framework of simple set theory.

The behavior description can therefore document several alternative solutions of the goal. The description of union, defined above, can be changed to encompass the new skill union-1 and the new performances obtained from the concrete implementations by adding the following items:

Skill: b. D = union-1(A,B)

Performance: 1. A = '(a b c) B = '(d e)
D = '(c b a d e) [skill = b, union-1, LISP]

3.4 Example 2: Set intersection.

The intersection of two sets A and B is the set of all elements of both A and B. The computation of this set can be viewed as the combination of two steps: the calculation of $C = B - A$ and then of $D = B - C$. The set C is composed of those elements of B which are not in A. The goal is to calculate those elements of B which are in A, and this can be obtained by removing all elements of C from B. The computation is illustrated graphically in figure 1. The behavior description of intersection is defined in table 2.

	Intersection (simple set theory)
Goal:	Construct $D : (\forall A)(\forall B)(D = A \cap B)$
Knowledge:	i. $(\forall A)(\forall B)(\forall D)(A \cap B = D) \leftrightarrow (\forall e)(e \epsilon A \wedge e \epsilon B \leftrightarrow e \epsilon D)$
	ii. $(\forall A)(\forall B)(\forall C)(\forall D)$ $((\forall e)(e \epsilon A \wedge e \epsilon B \leftrightarrow e \epsilon D)) \leftrightarrow$ $((\forall f)(f \epsilon C \leftrightarrow f \epsilon B \wedge f \not\epsilon A) \wedge$ $(f \epsilon D \leftrightarrow f \epsilon B \vee f \not\epsilon C))$
Skill:	a. C = diff(B,A), D = diff(B,C)
Performance:	1. A = '(1 2 3 4) B = '(4 3 5 6) D = '(4 3) [skill = a, set-intersection, LISP]

Table 2. The behavior description for intersection

As in the previous case, the knowledge defines an expansion of the goal into a form where skills can be defined. The skills, or, more precisely, their concrete level counterparts, actually carry out the computation and the performance is

recorded. Again, LISP, PROLOG and C versions of this computation were defined using the same abstract model.

4 The documentation of a configuration system

For the simple programs considered so far, it was possible to give a precise definition of the knowledge of set theory represented in the programs. In this section we show how our approach can be scaled up to describe a conventional KBS. We describe a KBS which solves the Sisyphus task of configuring elevator systems (Yost, 1992). Due to space considerations we present only a part of the entire solution.

The Sisyphus task involves the configuration of elevators according to a given design. The aim is to find a configuration which satisfies a number of constraints. We shall not describe the derivation of our solution, in fact we used an approach similar to the KADS methodology[5] (Breuker and Wielinga, 1989), instead we shall focus on the documentation of the solution.

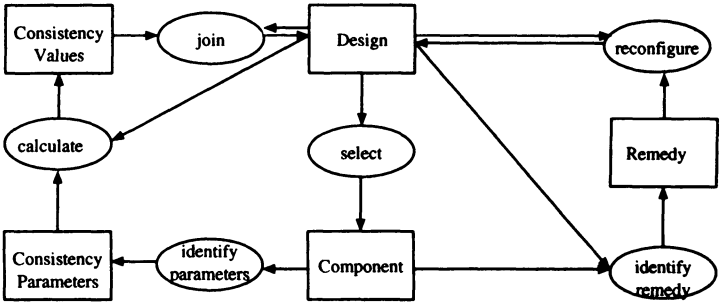

Fig. 2. An inference structure for configuration

A KBS will typically be comprised of function definitions which define the problem solving steps and a knowledge base (KB) which defines domain specific information. The function definitions will normally be task specific but domain independent, as will the schema of the KB. Our solution of the Sisyphus task has exactly this design. It was derived after a knowledge level analysis of the task was performed. The resulting inference structure is shown in figure 2. The method should be understood as beginning with the selection of a *Component* from the *Design*, determining a number of *Parameters* which determine the consistency of this choice, and their values. The checking of the consistency of the choice of component is done by the *id-remedy* knowledge source which finds a replacement component, *Remedy*, if any constraints are violated.

[5] We adopt the KADS I terminology

The documentation of the KBS must characterise two distinct types of knowledge: the domain knowledge of the KBS, and the method of the configuration algorithm. The knowledge about the attributes of a particular motor is represented in table 3. The Knowledge component is a formalization of the domain knowledge as arguments of the predicates *instance-of* and *has-attribute*. These predicates have the usual interpretation. The Skill component is the construction of a list where slots in the design are assigned the appropriate value. There is only one possible performance, the slots get the correct values.

The configuration algorithm consists of the recursive function **configure**, which is defined in terms of two further functions, **increment-design** and **identify-remedy**, one list processing function and functions defined in the KB. In order to document the **configure** function we must specify the goal which it satisfies:

Construct Design = configure(Specification)

We consider the knowledge of **configure** to be a formalized representation of the inference structure which resulted from the knowledge level analysis. The meta-classes of the inference structure are captured by predicates whose logical domains are sets of domain terms. The dependency of meta-classes upon one another is represented by implications, and the appropriate quantification of variables. For example, in our inference structure the knowledge source *select* selects a *Component* to be configured, given the configuration *Design*. *Component* and *Design* are predicates and the dependency is represented by the formulae:

i. $(\forall d : set)(Design(d) \rightarrow (\exists c : element)Component(c))$
ii. $(\forall c : element)(Component(c) \rightarrow (\exists d : set)Design(d))$

Formula i. states that for all designs a component can be selected, ii. states that if a component has been selected then a design must exist. These formulae define the key data classes of the domain and specify a number of consistency relationships which must hold. Each knowledge source of the inference structure is described in a similar way and the resulting set of formulae describe a set of relations which hold in the inference structure as a whole[6]. This method of describing the inference structure does not define how components are selected, or when, in control terms, the selection inference is made, these aspects are specified in the Skill component.

The Skill component is simply the ordered listing of the definition of *configure*. Details such as terminating conditions are omitted to leave the essential structure of the algorithm.

[6] This formalization of the knowledge level does not predict the results of the problem solving, it specifies necessary relationships between KL classes. The role of this formalization is descriptive and not predictive or computational. No stronger formalization of the KL is possible without introducing domain terms and/or control terms.

	Instantiate attributes (elevator configuration)
Goal:	Construct $Y : Y = instantiate - attributes(motor, 10HP)$
Knowledge:	i. $instance - of(10HP, motor)$ ii. $has - attribute(10HP, model, 10HP)$ iii. $has - attribute(10HP, maxHP, 10)$ iv. $has - attribute(10HP, maxCurrent, 150)$ v. $has - attribute(10HP, weight, 374)$
Skill:	a. Y = ((motor.model 10HP) (motor.maxHP 10) (motor.maxCurrent 150) ...)
Performance:	Y = ((motor.model 10HP), (motor.maxHP 10) (motor.maxCurrent 150) ...)

Table 3. Domain knowledge about the motor model 10HP

a. <Component,NewDesign> = increment-design(Design,Remedy),
NewValues = calculate-values(Component,NewDesign),
NextDesign = join(NewValues,NewDesign),
NewRemedy = identify-remedy(Component,NextDesign)
return <O-Design, O-Remedy> = configure(NextDesign,NewRemedy)
where Design, Remedy are input values,
and O-Design, O-Remedy are output values

The function increment-design returns a *Component* and an updated version of the design as the outputs. More details of this function are documented in a separate behavior description which shows that increment-design calls the instantiate-attributes function defined above. The relationship between the Skill and Knowledge components is indicated by the variable names, i.e. those which end in Design to be interpreted as instances of the *Design* meta-class.

The Performance component is an example of an execution of the *configure* function.

1. Design = (... (motor.model nil) ...)
   Remedy = nil
   NextDesign = (...(motor.model 10HP) (motor.maxHP 10) ...)
   NewRemedy = 15HP
   O-Design = (... (motor.model 15HP) (motor.maxHP 15) ...)
   O-Remedy = nil
2. Design = (... (motor.model 20HP) ...)
   Remedy = 28
   NextDesign = (...(motor.model 20HP) (machine.model 28) ...)
   NewRemedy = nil
   O-Design = (... (motor.model 20HP) (machine.model 38) ...)
   O-Remedy = nil

The documentation of the KBS differs from the documentation of the set

theory programs in that domain knowledge must be described. This is unproblematic. The knowledge of the algorithm, that is, the knowledge of the solution method implemented by the algorithm, cannot be expressed in such a way that the KL definition predicts the symbol level performance. This is a known feature of the knowledge level, the KL cannot be reduced to the symbol level. The set theory programs are exceptional in that the symbol level behavior is precisely that predicted by the Knowledge component, within the physical limitations of the computer which performs the calculation.

In contrast with the KADS approach, we have not specified an implementation where the inference structure is represented as a distinct layer of the design. Instead, we have produced a design which can be easily related to the inference structure and this simplified the implementation. The documentation of our solution includes examples of runs of the program, hence our approach is more empirical than the purely rationalistic KADS method.

5 Sharing and reuse

The idea of significance as related to context is important in our approach to the reuse of behavior descriptions. Consider, for example, that you observe an application of the **concatenate** function:

concatenate((()(1)(2)(2 1))((3)(3 1)(3 2)(3 2 1))) =
(()(1)(2)(2 1)(3)(3 1)(3 2)(3 2 1))

This function would just be described as concatenating two sets. However, if the following piece of information is added:

powerset((2 1)) = (()(1)(2)(2 1))

then we can see that the first argument of **concatenate** is the powerset of the set (2 1). If we now add two more pieces of information:

repeated-union((3),(()(1)(2)(2 1)) = ((3)(3 1)(3 2)(3 2 1)) and

split((3 2 1)) = <(3), (2 1) >

we can see that the second argument of **concatenate** is the list formed by adding (3) to the powerset of (2 1) and that these elements were once composed into a single list (3 2 1). Taken together these functions define an algorithm which computes the powerset of a set. The significance of the particular application of the **concatenate** function increases as more information about the context in which the evaluation occurs becomes known. We have, of course, only described the process of understanding the significance of the **concatenate** function in one particular instance, the problem of designing algorithms such as powerset is not addressed here.

In general, we view the significance of a computation as increasing as it becomes embedded in greater and greater contexts, i.e. as the system grows. The problem is to describe and redescribe the computation as this growth happens in order that reuse can occur. Behavior descriptions capture the information required for this purpose.

The complete behavior description of the powerset function is shown in table 4. The computation of this function can be defined recursively as is shown in

	Powerset (simple set theory)
Goal:	Construct $Y : (\forall A)Y = P(A)$
Knowledge:	i. $(\forall A)(\forall Y)(Y = P(A) \leftrightarrow (\forall x)(x \epsilon Y \leftrightarrow x \subseteq A))$ ii. $(\forall A)(\forall Y)(Y = P(A) \leftrightarrow$ $(\forall B)(\forall C)(\forall D)(\forall E)(\forall F)(C = A - \{B\}) \wedge (D = P(C)) \wedge$ $(\forall z)(\forall y)(z \epsilon D \wedge\ y = \{B\} \cup z \rightarrow y \epsilon E) \wedge (Y = E \cup D)$
Skill:	a. <B,C> = split(A), D = powerset(C), E = repeated-union(B,D), Y = concatenate(D,E)
Performance:	1. A = '(2 1), Y = '(()(1)(2)(2 1)) [skill = a, powerset, LISP] 2. A = '(3 2 1) Y ='(()(1)(2)(2 1)(3)(3 1)(3 2)(3 2 1)) [skill = a, powerset, LISP]

Table 4. The behavior description of powerset.

figure 3. The powerset Y, of set A, is the set of all subsets of A. The basic insight required to derive the computational method from the requirements is to note that the powerset of a set A can be calculated from the powerset of A minus element B and the set formed by adding B to every subset of D.

The method is to split one element, B, of the input set A, to obtain set C. The powerset, D, of C is calculated and then the set E is created by adding B to each member of D. The union of D and E yields the powerset of A. In fact, as D and E are disjoint (there are no members of B in D), the powerset can be obtained by the concatenation of D and E. This is documented in table 4.

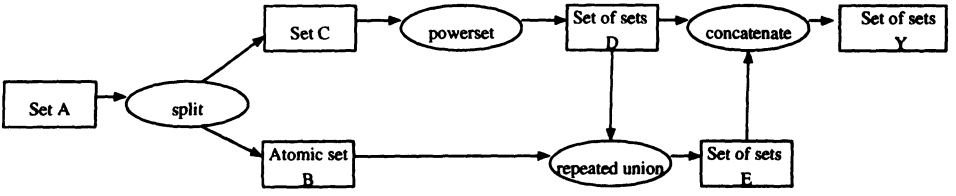

Fig. 3. The powerset computation.

6 Conclusions

We have defined behavior descriptions in order to overcome a number of problems with fixed ontologies. We have shown how behavior descriptions can be formalized and illustrated our proposals by a number of simple examples from set theory. The documentation of a knowledge-based system for the Sisyphus task demonstrates how our approach can be scaled up.

The knowledge level is a rational approach for describing the behavior of a computational system. In addition to this descriptive function, the knowledge level is also used in the development of KBS. Typically, declaratively represented knowledge plays a specific role in a problem solving model, however, the documentation of such a system relies on a fixed interpretation. In contrast, our approach is more evolutionary (Kühn, 1993) (Hinkelmann, Meyer and Schmalhofer, 1994). Behavior descriptions may change over time.

We do not separate the description of the requirements from the description of the implementation (Swartout and Balzer, 1982) and the system is not viewed as being divorced from the environment in which it is used. In comparison with the discussion of the knowledge level/symbol level relationship in Smith and Johnson (1993) we emphasise the notion of *method* in the knowledge level description.

From Clancey's discussion of the frame of reference problem we can conclude that there may be different descriptions for the same system depending upon context. Pursuing this argument, we consider it inevitable that systems will be redescribed for different contexts and purposes. This contrasts with the view of a fixed ontology based upon a uniform KL descriptions. For allowing such conceptual changes on KBS and programs, we proposed the behavior descriptions and provided a formalization which can be used for documentation. Such documentation consists of, 1) the goal or purpose the KBS is used for, 2) the distinctions and categorizations which are made by the system, i.e. representation of declarative knowledge, 3) the computational procedures which are implemented by the system and correspond to the categorization made by the encoded declarative knowledge and 4) the actually observed input-output relations (i.e. the performance of the system on selected test cases).

If the use of a system is not to be limited by the designer's preconceptions, then the abstract descriptions of a system have to be changed according to the new purposes which users may invent. We have shown how behavior descriptions are adjusted to new purposes by assessing the already existing behavior description in the light of the newly emerging conceptualization. It is often assumed that knowledge sharing and reuse is to be accomplished by specifying a (rather) fixed ontology, which is agreed upon by all designers and users. The proposed behavior descriptions provide the flexibility of describing and redescribing systems according to the current purposes. Behavior descriptions thus allow for the sharing and reuse of knowledge as well as for conceptual changes over time (Keil, 1993; Ram, 1993). Specific representation languages may be required for forming such descriptions. A declarative language which allows for knowledge evolution is currently being developed in the VEGA project (Boley, 1993).

We view reuse as a problem of the redescription of computational systems in

larger contexts. Components of a system gain significance by being embedded. An example of such an embedding is the reuse of the concatenate function within the powerset function. Another example is the reuse of calculations in a spreadsheet environment by embedding them in a larger workflow e.g. for paying travel expenses (see for example, Dallemagne *et. al.*, 1992). We have not yet presented a technology for reuse, but we have laid some of the theoretical groundwork for such a technology.

Acknowledgements: We would like to thank Otto Kühn, Peter Ossario, Jörg Thoben and Rainer Deventer for their critical and constructive discussions of the issues presented in this paper. This research was supported by grants ITW 8902 C4 and 413-5839-ITW9304/3 from the BMFT and by grant SCHM 648/1 from the DFG.

References

Anderson, J.R. (1983) The architecture of cognition. *Harvard University Press, Cambridge, Massachusetts, 1983*

Anderson, J.R. (1990) The adaptive character of thought. *Hillsdale, New Jersey, Lawrence Erlbaum, 1990*

Boley, H. (1993) Towards evolvable knowledge representation for industrial applications. *in Hinkelmann, K. and Laux, A. (eds) DFKI Workshop on Knowledge Representation Techniques, D-93-11, DFKI, July 1993*

Bourne, L.E. Jr.(1969) Concept learning and thought: Behavior not process. *in ed. Voss, J. Approaches to thought. Columbus OH: Merrill, :167-195.*

Bourne, L.E.,Jr., Ekstrand, B.R. and Dominowski, R. L. (1971) The psychology of thinking. *Englewood Cliffs, N.J.: Prentice-Hall, 1971.*

Breuker, J. and Wielinga, B. (1989) Models of expertise in knowledge acquisition. *In Guida, G. Tasso, C. (Eds.), Topics in expert system design, methodologies and tools Amsterdam, Netherlands: North Holland :265 - 295*

Chase, W.G. and Simon, H.A. (1973) Perception in chess. *Cognitive Psychology, 4, :55-81.*

Clancey, W.J. (1985) Heuristic classification. *Artificial Intelligence 27 (1985) :289-350*

Clancey, W.J. (1989) The frame-of-reference problem in cognitive modelling. *In Proceedings of the 11th Annual Conference of the Cognitive Science Society Hillsdale, NJ: Lawrence Erlbaum :107-114*

Clancey, W. J. (1991) The frame of reference problem. *in ed. VanLehn, K. Architectures for intelligence. Lawrence Erlbaum, New Jersey, 1991. :357-423*

Dallemagne, G. Klinker, G. Marques, D. McDermott, J. and Tung, D. (1992) Making application programming more worthwhile. *in Schmalhofer, F. Strube, G, and Wetter, T. (eds) Contemporary knowledge engineering. Springer Verlag, Berlin, 1992 :6-22*

Dietterich, T.G. (1986) Learning at the knowledge level. *Machine Learning 1:287-316 1986*

Fischer, G. and Girgensohn, A. (1990) End-User Modifiability in Design Environments, *Human Factors in Computing Systems, CHI'90, Conference Proceedings (Seattle, WA) ACM, New York (April 1990), :183-191.*

Gruber, T. (1993) Towards principles for the design of ontologies used for knowledge sharing. *ftp server Stanford*

Guha, R. V. and Lenat, D.B. (1990) CYC: A mid-term report. *AI Magazine 11(3) :32-59*

Hinkelmann, K. Meyer, M. and Schmalhofer, F. (1994) Knowledge-base evolution for product and production planning. *AICOM Vol. 7 Nr. 2 :98-113*

Keil, F.C. (1993) Conceptual change and other varieties of cognitive development: Some distinctions in the emergence of biological thought. *Proceedings of the Conference of the Cognitive Science Society, Boulder, Co. 1993, Lawrence Erlbaum, NJ:4*

Kühn, O. (1993) Knowledge sharing and knowledge evolution. *Proceedings IJCAI 1993 Workshop on knowledge sharing and information exchange.*

Maes, P. (1993) Behavior-Based Artificial Intelligence. *in Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society, June 18-21, 1993, :74-83. Hillsdale, NJ: Lawrence Erlbaum.*

Miller, G.A. (1956) The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review, 63 :81-97.*

Musen, M. A. (1992) Dimensions of knowledge sharing and reuse. *Computers and biomedical research 25, 435467 (1992)*

Neches, R. Fikes, R. Finin, T. Gruber, T. Patil, R. Senator, T. and Swartout, W.R. (1991) Enabling technology for knowledge sharing *AI Magazine Fall 1991 :36-56*

Newell, A. (1969) Discussion of Professor Bourne s Paper: Thoughts on the concept of process. *in ed. Voss, J. Approaches to thought. Columbus OH: Merrill, :196-210.*

Newell, A. (1982) The knowledge level. *AI 18 (1982) :87-127*

Newell, A. (1990) Unified theories of cognition. *Cambridge, MA: Harvard University Press.*

Newell, A. (1992) Precis of Unified Theories of Cognition. *Behavioral and Brain Sciences, 15, :425-437.*

Norman, D.A. and Bobrow, D.G. (1975) On data limited and resource limited processes. *Cognitive Psychology, 7, :44-64*

Pylyshyn, Z. (1984) Computation and cognition. *Cambridge, MA: MIT Press.*

Ram, A. (1993) Creative conceptual change. *Proceedings of the Conference of the Cognitive Science Society, Boulder, Co. 1993, Lawrence Erlbaum, New Jersey :17-26*

Ryle, G. (1949) The concept of mind. *Harmondsworth: Penguin*

Schmalhofer, F. and Thoben, J. (1992) The model-based construction of a case-oriented expert system. *AICOM Vol.5 Nr.1 :3-18*

Schreiber, G., Akkermans, H. and Wielinga, B. (1990) On problems with the knowledge level perspective. *in. Boose, J. Gainers, B.R. (eds) Proceedings of the 5th Banff Knowledge-bases systems workshop, :30-1 -30-14.*

Simon, H.A. (1974) How big is a chunk? *Science, 1974, 183, :482-488*

Smith, J.W. and Johnson, T.R. (1993) A stratified approach to specifying, designing, and building knowledge systems. *IEEE Expert, 8(3) :15-25*

Squire, L.R. and Slater, P.C. (1975) Forgetting in very long-term memory as assessed by an improved questionnaire technique. *Journal of Experimental Psychology: Human Learning and Memory, 1975, 1, :21-34*

Sticklen, J. (1989) Problem solving architecture at the knowledge level. *Journal of Experimental and Theoretical Artificial Intelligence.*

Swartout, W. and Balzer, R. (1982) On the inevitable intertwining of specification and implementation. *Communications of the ACM, 25, 7 :438-440.*

Swartout, W.R. Neches, R. and Patil, R. (1993) Knowledge sharing: Prospects and challenges. Proceedings of the International conference on building and sharing of very large-scale knowledge bases 1993 :95-102

Tulving, E. (1975) Ecphoric processes in recall and cognition. *in Brown, J. (ed) Recall and recognition. Wiley, 1975*

Winograd, T. (1976) Frame representations and the declarative-procedural controversy. in Bobrow, D.G. Collins, A. (eds) Representation and Understanding. *New York: Academic Press, :185-210.*

Yost, G. (1992) Configuring elevator systems. *AI Research Group, DEC, Marlboro, MA.*