Chapter 2

Representation and Management of Knowledge

Knowledge, its representation and its management play a central role in Artificial Intelligence. However, it cannot be said that the views and approaches in this field are converging. As Brachman and Levesque [1985] pointed out in the introduction of an anthology of papers on knowledge representation, there is no textbook available on this subject, and, probably, it would be impossible to write one at this point.

The following sections certainly cannot compensate for this deficiency. However, what I will try to achieve is a presentation of my basic assumptions about what knowledge representation amounts to. These assumptions can be roughly characterized as the *logical point of view.* Additionally, in the last section, I will briefly relate my work to what is called *Knowledge Base Management Systems.*

2.1 Knowledge and its Representation

Research in Artificial Intelligence (henceforth AI) started off by trying to identify the *general mechanisms* responsible for intelligent behavior. One instance for this line of research is the General Problem Solver[1] GPS [Ernst and Newell, 1969]. The main paradigm was that "One had not to 'cheat' by 'giving' the problem-solver the solution in any sense" as Hayes [1974] characterized it. However, it quickly became obvious that general and powerful methods are not enough to get the desired result, namely, intelligent behavior. Almost all tasks a human can perform which are considered to require intelligence are also based on a huge amount of knowledge. For instance, understanding and producing natural

[1]This system is not only an example of the mentioned paradigm, but also a very striking instance of what D. McDermott [1976, p. 4] calls the fallacy of "wishful mnemonics": "...it originally meant 'General Problem Solver,' which caused everybody a lot of excitement and distraction. It should have been called LFGNS – 'Local-Feature-Guided Network Searcher'."

language heavily relies on knowledge accumulated during our life time, knowledge about the language, about the structure of the world, social relationships etc. Therefore, it seems to be inevitable that this knowledge must be utilized in some way.

The most straightforward solution would be that the programmer of an AI system just "programs the knowledge into the system," using an AI programming language, such as LISP [Winston and Horn, 1981] or PROLOG [Clocksin and Mellish, 1981]. The outcome of this effort would be a system based on the knowledge of its programmer, a "knowledge-based" system. However, something seems fundamentally wrong with this solution: we are giving up on generality. Although the program might exhibit an interesting intelligent behavior in some special field, we get no hints of how general the solution is, whether it can be moved to other application areas, or how it relates to intelligence (whatever that may be). We – and the system as well – may even be unable to explain which knowledge is responsible for a particular system behavior. Thus, such an approach precludes the realization of something like an *explanation facility* for such a system, a capability often explicitly required for complex systems [Wahlster, 1981]. Furthermore, there seems to be no difference from other programming activities in computer science.[2]

Of course, knowledge in AI is utilized in a different way, and here *knowledge representation* (henceforth KR, or simply representation) comes into play. Knowledge is not just programmed into a system, but *explicitly* and *declaratively represented* by using some kind of formal language – a *KR formalism.* Explicit means that the system architecture contains an *explicit knowledge base* (henceforth KB), which, in turn, contains expressions of the KR formalism. These expressions describe in a (more or less) declarative way the knowledge the system relies on to solve a given problem. Only if these conditions are met, we call a system *knowledge-based* [Brachman and Levesque, 1985, p. xiv].

This characterization implies a couple of other properties a KB and a knowledge-based system must possess. However, it also leaves some questions unanswered, namely:

- What is the relationship between knowledge and represented knowledge? This includes a question of what the *nature of knowledge* is (from the point of view of AI).

- What are the *necessary properties* of KR formalisms? Stated in another way, is any formal language a KR formalism?

- What does *declarativity* mean, or, stated differently, where do expressions get their *meaning* from?

[2] Although such a view is possible [Stoyan, 1987], I believe that there are a number of differences, which hopefully become obvious in this chapter.

- What are the *services* a system manipulating represented knowledge should provide?
- What are the appropriate *techniques* to implement a system dealing with represented knowledge – a *KR system*?

2.1.1 Knowledge and Data

One way of approaching the question of what knowledge is might be to contrast the notion of *data* as understood in the data-base field with *knowledge* as used in AI. Wiederhold [1986, p. 79] characterizes the difference in the following way:

> If we can trust an automatic process or clerk to collect the material then we are talking about *data*. ...
>
> If we look for an expert to provide the material then we are talking about *knowledge*. Knowledge includes abstraction and generalization of voluminous data. ...

This point of view, however, is influenced by the systems Wiederhold [1986, p. 78] intends to describe, namely, information systems "to be used for decision-making in enterprises". Furthermore, the characterization seems to be influenced by one very prominent kind of AI systems: expert systems [Waterman, 1986]. Even though it often might seem that AI consists mainly of research in this field, in particular if AI is seen from the outside, representation of expert knowledge is only one topic. For instance, AI and KR are very much concerned about the representation of *common-sense* knowledge [Hobbs and Moore, 1985], which, apparently, is possessed not only by experts. Additionally, as Wiederhold himself admits, the definition is only of a pragmatic nature. The distinction between what knowledge is and what data is may change according to who is interpreting the information.

Wong and Mylopoulos [1977, p. 344] tried to characterize the difference indirectly by specifying what is stored in knowledge bases and data bases. Knowledge bases will

> include large amounts of abstract knowledge ... and a small amount of (less interesting) concrete knowledge ...

while data bases

> were designed mostly for concrete knowledge, and the modelling methods were inadequate for representing abstract knowledge.

Here again, we meet the distinction between raw data (by the way, what does *raw* data really mean?) and abstractions. However, it is clearly spelled out that both are considered as some kind of knowledge.

Obviously, it does not give a satisfying answer to our question. The only hint we got is that data stored in data bases could be viewed as represented knowledge, albeit a very limited sort. All in all, it does not seem that this small excursion into the data-base field contributed to any deeper understanding of knowledge or how it could be represented.

2.1.2 The Knowledge Level

If in AI the term *represented knowledge* is used, it most often refers to the expressions of a KR formalism. In addition to that, of course, there must be some processes using the KR expressions which assign the (structural, internal) meaning to the expressions [Barr and Feigenbaum, 1979, p. 143]:

> We can say, metaphorically, that a book is a source of knowledge, but without a reader, the book is just ink on paper. Similarly, we often talk about of the list-and-pointer data structures in an AI data-base as knowledge per se, when we really mean that they represent facts or rules when used by a certain program to behave in a knowledgeable way.

This leads to a totally different perspective on knowledge and data than the one in the last subsection, namely, something like

$$Knowledge = Data + Interpreter$$

This sounds simple and plausible, but it is quite misleading. Consider, for instance, two different Data/Interpreter pairs. Are they necessarily different bodies of knowledge? Even if they behave identically? Certainly not.

Knowledge does not seem to be describable in mechanical terms. It seems to be a property we ascribe to an agent when it is acting. And this ascription is not just arbitrary or accidental, but rather serves as an important abstraction which permits the explanation of behavior. If we observe someone we consider as a *rational agent*, we may explain the agent's actions in the following way [Newell, 1982, pp. 105]:

1. The rational agent has some *knowledge* about the environment and how it could be affected by actions.

2. The rational agent has some *goals* (also a kind of knowledge).

3. Then the agent's actions are based on the knowledge and aimed at achieving some of the goals.

This explanation does not give us any predictive power because we do not know what will happen in the case of conflicting goals, when goals can be achieved by different actions, or in the case of inaccurate knowledge. It is more a competence-like notion. Nevertheless, it permits us to characterize knowledge in an important way [Newell, 1982, p. 105]:

> Knowledge is to be characterized entirely *functionally*, in terms of what it does, not *structurally*, in terms of physical objects with particular properties and relations. This still leaves open the requirement for a physical structure that can fill the functional role.

Newell [1982, p. 99] goes as far as postulating an additional computer system level distinct from all other levels (as the symbol level, the register-transfer level, etc.) – the *knowledge level*:

> There exists a distinct computer system level, lying immediately above the symbol level, which is characterized by knowledge as the medium and the *principle of rationality* as the law of behavior.

Although this level has some very weird properties compared with other computer system levels – it is only approximative, there exist no laws of composition, etc., and therefore its existence may be arguable – it does give us the opportunity to analyze knowledge at a very abstract level. Furthermore, KR systems can also be analyzed at this level. Instead of going into the details of how knowledge is represented symbolically in a particular KR system we may just view it on the knowledge level and ask how it behaves on particular inputs. This enables us, for instance, to compare such different representation formalisms as *semantic networks* as described in Sect. 3.1.1, *frame systems* as sketched in Sect. 3.1.2, *production-rule systems*, *relational data base system*, etc. Certainly, we cannot perform the comparison without an appropriate analytical tool. Newell [1982, p. 121] suggests for that purpose logic:

> Just as talking of *programmer-less* programming violates truth in packaging, so does talking of *non-logical analysis* of knowledge.

If anybody has the feeling that, besides the fact that the term *knowledge* was used very often above, the ideas sound very familiar and similar to *abstract data types* [Liskov and Zilles, 1974], he or she is not completely misled. As a matter of fact, Levesque [1984a] describes KBs on the knowledge level in terms of abstract data types using two operations: TELL to add more knowledge to a KB and ASK to query the KB.

2.1.3 The Knowledge Representation Hypothesis

In agreeing with the above point of view, we may arrive at the question of how it is possible to find a way to *represent* knowledge, this rather abstract object. Although there is, or at least was, a broad diversity of opinions of what representation of knowledge amounts to, as Brachman and Smith [1980] concluded from a questionnaire, there seems to be at least one common assumption, namely, that "knowledge is representational." That means first, if we "know something," then there is an object of our knowledge, i.e., we know something *about some particular entity*.[3] Second, this knowledge can be in some way symbolically encoded,[4] and if the knowledge is applied then the symbols are manipulated without regard to the entities the symbols refer to. This point of view is spelled out by Brian C. Smith [1982, p. 2] as the *Knowledge Representation Hypothesis*:

> Any mechanically embodied intelligent process will be comprised of structural ingredients that a) we as external observers naturally take to represent a propositional account of the knowledge the overall process exhibits, and b) independent of such external semantical attribution, play a formal but causal and essential role in engendering the behavior that manifests that knowledge.

Whether we subscribe to this point of view in the strong version – *any* knowledge is representational – or in a weak version – knowledge is *possibly* representational – or whether we deny both as, for example, Dreyfus [1981] does,[5] we have to admit that

- AI and the paradigm of KR has some success. Expert systems, a now very popular species of knowledge-based systems, proved that KR can be applied successfully.

- If we want to build "smarter computer systems" – one goal of AI [Winston, 1984, p. 3] – it seems that we really do not have a better hypothesis today than Smith's.

- Even if the hypothesis may be wrong for the human as a whole, the analysis of mind – another goal of AI – can rest on this hypothesis, provided that some significant aspect of thought can be understood in this way [Haugeland, 1985, pp. 246–254].

To summarize, the knowledge representation hypothesis seems to be reasonable.[6] If we accept the hypothesis as a base for further investigation (and we will do so in the following), this yields some consequences. An important

[3] Entity here is meant to be a very general category. It could be an existing object in the world, relationships between objects, nonexistent objects, ideas, events, etc.

[4] This is not a commitment to any particular kind of representation, e.g. propositional or analogical, or whatever [Hayes, 1974; Levesque, 1986], as Schefe [1982, p. 43] seems to assume.

[5] Dreyfus denies that knowledge *can* be representational. He argues that intelligence instead depends on skills acquired by practice and training and that this cannot be formalized.

[6] Furthermore, no empirical evidence or arguments, in particular by connectionists [Rumelhart and McClelland, 1986], show that the hypothesis is mistaken. On the contrary, it seems likely that connectionism will have to adopt organizing principles that resemble the knowledge representation hypothesis [Chandrasekaran *et al.*, 1988].

one is that KR is a kind of *reconstruction* of human knowledge.[7] We do not directly represent the human knowledge by a mapping from "objects of the mind" to expressions of some KR formalism (in the first place), but instead we try to establish a system which is *functionally similar* to human knowledge.

2.1.4 Three Approaches to Knowledge Representation

In the extreme, a KR formalism could be viewed as "mentalese" (as B. C. Smith [1982, p. 3] termed it), the language our mind uses. Of course, nobody holds this view. We are just too far away from a complete understanding of what is going on in our brain.

Nevertheless, in KR, as well in AI in general, there are at least two different intentions in working out solutions. One is to build performance systems, i.e., systems which act intelligently. In this case, only engineering principles are important in building a system (let us call it the engineering perspective). The other intention is to get insights into the mind. Here, the physical structure of a process is much more important for the realization of a solution than before. In fact, the structure of the system is considered to be more important than the system's being used for some purpose (this is usually called the cognitive perspective).

Examples of these two different approaches are HAM-RPM [von Hahn *et al.*, 1980], a natural language system which was used to investigate the cognitive abilities of an agent engaging in a natural language dialog, and HAM-ANS [Hoeppner *et al.*, 1983], a natural language access system to different background systems. These two systems present also a good example for the fact that even though the goals seem to be fundamentally different both approaches are fertilizing each other: HAM-ANS is the descendant of HAM-RPM. Another good example of cross-fertilization in the field of KR are production-rule systems. In the early days of AI, they were used as a tool for demonstrating some cognitive properties; later on, they were employed by expert systems [Davis and King, 1984].

The reasons for this cross-fertilization are obvious. In trying to build an AI performance system, we have to create at least a human-friendly interface, often resulting in a human-like interface. In a natural language system, for instance, we have to account for misunderstandings [McCoy, 1984]. Furthermore, instead of interpreting a question literally, we had better infer the intention of the human questioner and answer appropriately [Marburger and Wahlster, 1983].

In research oriented around cognition, on the other hand, it is known that the human mind does extraordinarily well on cognitive tasks, so that a good engineering solution might show us the way to the structure we are looking for. An example of this, at least partly, is the work of Winograd [1972]. Of course, an

[7] An observation also mentioned by Schefe [1982] leading him to coin the term *knowledge reconstruction*. We, however, will stick to the established terminology and will continue to use the term *knowledge representation*.

efficient solution is not enough. Often, timing considerations and the ability to produce human-like failures are also important in evaluating a solution [Christaller, 1985, pp. 9ff].

Returning to our subject – knowledge representation – we can identify another perspective which does not fit into the scheme established so far. I would like to call that the epistemological perspective [McCarthy and Hayes, 1969; McCarthy, 1977]. The focus of research guided by this perspective is neither to build performance systems nor to investigate cognitive structures, but to develop a *theory of common-sense knowledge* and *reasoning* based on well-founded formalization, which might just be viewed as pursuing the program Newell proposed in his paper, namely, to analyze knowledge at the knowledge level. That does not mean that this line of research ignores the other goals, or that it is useless for the other perspectives. On the contrary, by formalizing the ideas which are used in existing AI systems, we get a better understanding of what these systems really do (or are supposed to do) – and this seems to be one important way to develop AI as a scientific discipline [Hobbs and Moore, 1985, p. xxi] – and it opens up the opportunity, for instance, to analyze AI systems with respect to correctness and computational complexity.

The research reported here is mostly motivated by goals of the latter perspective. This, however, is not the entire story. We are also aiming at developing the appropriate implementation techniques (at the symbol level) and demonstrate their feasibility by a running prototype. However, we will mostly ignore all issues which could be summarized under the heading of "cognitive adequacy" of the system structure.

2.2 Knowledge Representation Formalisms

While the characterization of knowledge as a competence-like notion in the last section seems to be the most plausible one, of course, the "functional role" has to be filled by a "physical structure," as Newell termed it. The *Knowledge Representation Hypothesis* tells us that this is possible.

What we need is some kind of formal language – a *KR language*[8]. Although any formal language might be used for this purpose, we had better assign meaning to expressions of such a language, in which case we will call the language a *KR formalism.*[9] A set of well-formed expressions[10] of such a KR formalism, which is intended to represent aspects of the world, will be called *knowledge base.*

[8] Called *scheme* by Hayes [1974].

[9] I do not intend to give a broad survey of different KR formalisms here, but rather to view KR formalisms from a very abstract point of view. For an overview of particular KR formalisms, the reader should consult [Mylopoulos and Levesque, 1984] or [Barr and Feigenbaum, 1979, pp. 141–222].

[10] Hayes [1974] calls such expressions *configurations.*

Additionally, we need some kind of machinery which can make use of the expressions in the KB – a *KR system*. Such a system has to fulfill two tasks. First, it has to interpret the KB and turn it into a body of knowledge – in terms of the knowledge level – by *inferring new expressions* which are only implicit in the expressions stored in the KB. Without this ability, a KR system would be just a plain, unstructured memory.[11] We could only retrieve the expressions put into the KB previously. The second task is to provide *services to the outside world*, like e.g. TELL and ASK, which operate on the KB and make use of the inference capabilities of the KR system.

Although we cannot have a KR system without a KR formalism (or at least this is hard to conceive) and it does not seem to make much sense to talk about a KR formalism if there is no system which makes use of it, we can, of course, view them in isolation. A prerequisite is that we can talk about the *meaning* of expressions in a KR formalism without resorting to a particular system which uses the expressions.

2.2.1 Semantics of Representation Formalisms

We can, of course, manage explaining what the expressions of a KR formalism are intended to mean by giving an informal description and some examples. Perhaps, a running system can give even more insights. In fact, research in AI starts off using this experimental, engineering approach most of the time. However, staying on the informal side has several drawbacks.

One disadvantage of the informal way is that it leads to numerous arguments of the kind "... but my understanding of this example is different ..." Although a corresponding KR system will give a definite answer when tried on the expression, this situation reveals that we did not really understand what the system does and what expressions in the formalism mean. In scientific discourse, in particular, the lack of formal semantics is "a regrettable source of confusion and misunderstanding" [Hayes, 1974, p. 64] – and makes the comparison and analysis of different KR formalisms impossible.

Connected with this deficiency is the fact that it is impossible to say whether the KR system correctly implements the intended behavior. This might not be disturbing if a KR system is used to simulate cognitive abilities, but cannot be tolerated if a KR system is employed as a subsystem of a knowledge-based system for some application. In this case, it is mandatory to be able to predict the system's behavior. The only way to achieve that without resorting to the actual program code is to employ some sort of *formal semantics*.

There are different choices for assigning formal meaning to a KR formalism. We may, for instance, explain the meaning of a KR formalism by referring to the

[11] Note that even data base systems are more powerful by virtue of associated query languages and the employment of the closed world assumption.

processes – possibly abstract ones – which operate on the expressions in the KB, as for example is done in [Shapiro and Rapaport, 1986, p. 282]:

> Arcs . . . only have meaning within the system, provided by node- and path-based inference rules (which can be thought of as procedures that operate on arcs).

However, this *operational semantics* is, of course, not a very enlightening approach. It does not explain why "arcs" have to be used at all or what they stand for. Another similar approach is the "empirical semantics" described in [Reimer, 1985] and [Reimer, 1986]. Here, the well-formedness conditions for expressions in a KB and operations on them are specified with the aid of the formal specification language META-IV [Bjørner, 1980]. Although this gives us a formal and rigorous specification of the KR system, it leaves open what the expressions in the KB refer to. Both these approaches to formal semantics are *internal* in that they refer to the formalism and to the system but not to the outside world.

As has been pointed out by many researchers in knowledge representation [Hayes, 1977; McDermott, 1978; Moore, 1982; Patel-Schneider, 1987b], the only satisfying approach to assigning meaning to KR formalisms is taking seriously the ideas spelled out in Sect. 2.1.3, namely, that knowledge is about entities in the world. That means expressions of a KR formalism should derive their meaning in some way from reference to the external world; and the adequate tool for such a task is Tarskian model theory [Tarski, 1935].

The usual semantics for first-order predicate logic (see [Genesereth and Nilsson, 1987] or any standard introductory text to logic) is the canonical example of this kind of semantics. A structure is assumed, containing a set of objects, called the *domain* or *universe of discourse*, functions on and relations over this domain, plus an *interpretation function* which maps constants and variables into objects, function symbols into functions, relation symbols into relations, and propositions into truth values. Any such structure (usually called *interpretation*) satisfying a set of propositions is called a *model* of this set of propositions, and, preferably, the (aspect of the) world we intend to describe with a set of propositions should be one of all the possible models.

As a matter of fact, it is this commitment to model-theoretic semantics which makes up the essence of the *logical point of view* in knowledge representation. It is neither a claim that a particular syntax has to be used to represent knowledge, nor a consensus about a particular logic formalism, nor even a restriction to sound inference rules [Israel, 1983], but it is the idea that the meaning of a formalism has to be derived by reference to the external world – by specifying an *external* semantics.

This is not a mere philosophical position irrelevant for "real work," although a large fraction of research in knowledge representation has seemed to get along without bothering about logics or semantics.[12] On the contrary, logic proved

[12]However, in recent years these issues appeared to be considered more important, as the

to be quite fruitful in the development of KR formalisms and systems. For instance, the development of the NETL system [Fahlman, 1979] – a semantic network system intended to run on a massively parallel computer – which started off using an operational semantics later turned to logic in order to eliminate some serious deficiencies [Touretzky, 1986]. The same holds true for the KL-ONE system [Brachman and Schmolze, 1985], which evolved over the years from a set of loosely coupled, intuitively justified procedures [Brachman, 1977] to a formalism with a well-defined semantics (as we will see in Sections 3.1.4 and 3.2) and formally justified inference procedures. In fact, in both cases the understanding of the representation formalism was considerably strengthened by the formal semantics.

Of course, the logical point of view is not the only possible one. There are a number of good arguments challenging this standpoint (the most important papers are [Minsky, 1975] and a recent article by D. McDermott [1987a]). We will, however, not go into the details of this discussion here (cf. Sect. 3.1.2).

Summarizing, the answer to one of the questions in Sect. 2.1 is that any formal language may be used as a KR formalism as long as a suitable semantics is assigned. This, of course, is not the entire story. There are also issues of expressiveness, convenience, and other criteria.

2.2.2 Adequacy Criteria

The most important criterion a KR formalism should fulfill is what McCarthy and Hayes [1969, p. 466] call *epistemological adequacy*[13]. This means it should be possible to *represent* all the aspects of the world we are interested in. This is, of course, a matter of degree, dependent on the purpose we are using the KR formalism for.

For example, a relational data base language [Date, 1981] is epistemologically adequate to represent a library catalog, but we might run into problems if we aim at representing concepts and relationships as in Sect. 1.3 with the same language. Although one can map a language as used in Fig. 1.1 (after a suitable formalization) into the language of relational data bases, it does not help very much. The semantics of relational data base languages is totally different from the semantics of the informal concept description language. Thus, we may *store* (in an abstract sense) concept descriptions using a relational data base language, but we cannot *represent* concepts with such a language.

Although epistemological adequacy seems to be enough since anything we want to represent can be represented, different KR formalisms can offer different degrees of convenience. Michie [1982] has coined the term *human window* in order to describe this degree of convenience. KR formalisms located within the human window are those which permit to use expressions mirroring the conceptualization

papers in the Knowledge Representation sections in proceedings of AI conferences prove.

[13]Woods [1983, p. 22] calls it *expressive adequacy*.

of the domain in order to represent something. Of course, employing Tarskian semantics helps a lot in staying inside the human window because the semantics forces us to assign meaning to expressions and sub-expressions by referring to the external world. Nevertheless, some facts may be expressed more easily and more naturally in one language than in another.[14] For example, the fact that a Small-team has at most 5 members is expressed easily in the informal concept description language in Fig. 1.1, but requires a couple of quantifiers and formulas if expressed in first-order predicate logic.

Another criterion for a KR formalism is its *heuristic adequacy*[15] [McCarthy and Hayes, 1969, p. 466] – its suitability to be used internally by a KR system to solve problems. Again, this is a matter of degree, mainly dependent on the algorithms the KR system uses. However, choosing the right formalism might enhance the performance of a system considerably. This is in particular the case if for a restricted problem domain a restricted KR formalism is chosen.

For example, first-order predicate logic is judged to be the epistemologically most adequate formalism for a wide range of representation tasks by many researchers [McCarthy, 1968; Hayes, 1977; Moore, 1982]. However, there are severe computational problems. First-order predicate logic is only semi-decidable, and, even for relatively simple cases, a theorem prover can spend a lot of time solving a problem. On the other hand, a restricted formalism can be more manageable, and problems which are hard to solve with the general formalism may turn out to be easily computable [Levesque and Brachman, 1985, p. 67]. Of course, one gives away general epistemological adequacy. However, we may aim at restoring this property by combining different formalisms – creating *hybrid representation formalisms*.

2.2.3 Hybrid Formalisms

Although it would be desirable to develop one KR formalism which meets all criteria – and in fact, some researchers are convinced that first-order predicate logic is just that formalism [Kowalski, 1980] – nowadays there seems to be a consensus that it is almost impossible to fulfill all goals simultaneously with one formalism. Sometimes it is worthwhile to switch between different KR formalisms because of heuristic adequacy [Sloman, 1985], as in the hybrid system CAKE [Rich, 1982; Rich, 1985]; sometimes the limited epistemological adequacy (which may come together with high notational efficacy) may suggest the combination of different subformalisms – making up a *hybrid* KR formalism.

The informal example given in the Introduction demonstrates the latter case.

[14]Woods [1983, p. 23] uses the term *conciseness of representation* to describe this dimension of a KR formalism.

[15]This criteria is called *computational efficiency* by Woods [1983, p. 22], and he subsumes this notion together with conciseness of representation and ease of modification under the heading *notational efficacy*.

The language used in Fig. 1.1 is very well-suited for the representation of terminological knowledge from the point of conciseness and heuristic adequacy but limited in its expressiveness. In order to say something definite about the world, we need an additional formalism to make assertions.

However, the mere combination of formalisms does not necessarily result in a hybrid formalism. A kind of "glue" is needed in order to constitute a hybrid formalism consisting of

- a *representational theory* (explaining what knowledge is to be represented by what formalism) and
- a *common semantics* for the overall formalism (explaining in a semantically sound manner the relationship between expressions of different subformalisms).

The representational theory should explain why there are different subformalisms, what their benefits are, and how they relate to each other. An answer should at least refer to the adequacy criteria introduced above:

- Epistemological adequacy, i.e., that the subformalisms are necessary to represent epistemologically different kinds of knowledge (e.g. analytic and synthetic knowledge).
- Heuristic adequacy, i.e., that the different subformalisms permit representation of the same knowledge in different ways for reasons of efficiency.

A necessary precondition for gluing things together is that their shapes fit, a fact which might be violated in designing a hybrid formalism, at least in the case where the subformalisms are intended to represent epistemological different kinds of knowledge. For example, if one subformalism permitted definition of terms by using time relationships but none of the other subformalisms referred to time at all, the subformalisms would be in some sense *unbalanced*. This, however, can be easily detected by inspecting the common semantics. However, there are also more subtle ways in which subformalisms can be unbalanced – a point we will return to in Sect. 4.6.

2.2.4 Formalizing a Body of Knowledge

Related to the problem of designing and implementing KR formalisms is the task of formalizing a body of knowledge, for example, the common-sense knowledge about fluids [Hayes, 1985]. It might seem that for a first attempt to formalize a new idea a very general KR formalism, the most epistemologically adequate formalism – say, first-order or higher-order predicate logic – would be a good candidate. There are some tradeoffs, though. Hobbs summarizes the issues involved as follows [Hobbs and Moore, 1985, p. xviii]:

> If only a narrow set of options is exercised in an unconstrained language, a more constrained language may be more natural and allow knowledge to be expressed more succinctly. In addition, special-purpose notations may indicate more explicitly the special-purpose deduction procedures that will use them. As Hayes admits, "idiosyncratic notations may sometimes be useful for idiosyncratic subtheories." Our concern in this volume is for facts, not notation, but sometimes getting the facts right requires notational maneuvers.

While I would admit that it might be distracting to make the move from a general to a special-purpose KR formalism too early, I also believe that the enterprise of formalizing a given domain of knowledge is not worth the effort it requires if we do not get more insights about the structure of the domain than a collection of axioms; particularly so, if no insights about the computational properties are gained.

This might sound a little overstated, and, in fact, I am not going to demand such results instantaneously, but the ultimate goal is to get the computer reason with the formalized knowledge, and this is impossible with unconstrained formalisms (as higher-order predicate logics). In order to give a positive example, the formalization of time as conducted by Allen [1983] and the analysis by others [Vilain and Kautz, 1986] represents a major leap in the right direction.

2.3 Knowledge Representation Systems

A KR system is in some sense the "materialization" of a KR formalism – it turns the semantics of the KR formalism into computations. However, KR systems do not exist in isolation. They are usually part of larger systems – of *knowledge-based systems*.

2.3.1 The Architecture of Knowledge-Based Systems

Analyzing existing knowledge-based systems, we note that large fractions of these systems do not have anything to do with "represented knowledge." For example, in [Bobrow *et al.*, 1986] it is reported that 30-50% of the program code in expert systems is devoted to the user interface. Even if we ignore the user interface, we notice that a KR system is usually only a subsystem of a larger system which is implemented using conventional, procedural programming techniques.[16] There are many reasons for this.

First of all, it is often more efficient to express knowledge procedurally,[17] in

[16] This does not mean that conventional programming languages are employed, but that the algorithmic paradigm is used in implementing the system.

[17] We could talk about "procedural knowledge representation" in this case. We will not do so, however. It is just programming – no more, no less.

particular if the anticipated use is obvious, rather than to use a general, declarative KR formalism. While this pays off in terms of efficiency and is often the only way to create practical systems [Hoeppner *et al.*, 1983], it restricts the ways the knowledge can be used, and it makes modifications difficult.

Second, the theory about particular phenomena is often developed by experimenting, which may involve the formulation of this theory using procedural schemes. Third, sometimes it is hard to find a declarative representation such that it is possible to specify a clean semantics which allows the important inferences to be drawn [McDermott, 1987a]. Most often, however, we will find a mixture of both kinds of expressing knowledge. Special-purpose theories are coded procedurally, but the parts which may often enough be changed and are well-understood from a formal point of view are represented declaratively.

Thus, a KR system has to offer services to the embedding knowledge-based system in some way. Furthermore, for some people – usually called *knowledge engineers* – it should be possible to manipulate the KB from the outside world[18] in order to "put knowledge into the KB." This could be done by directly manipulating data structures, by using a special KB editor, or by employing a separate knowledge acquisition system.[19] Summing up, the principal architecture of a knowledge-based system looks like Fig. 2.1 – characterizing the situation completely from a conceptual point of view, abstracting away organizational matters.

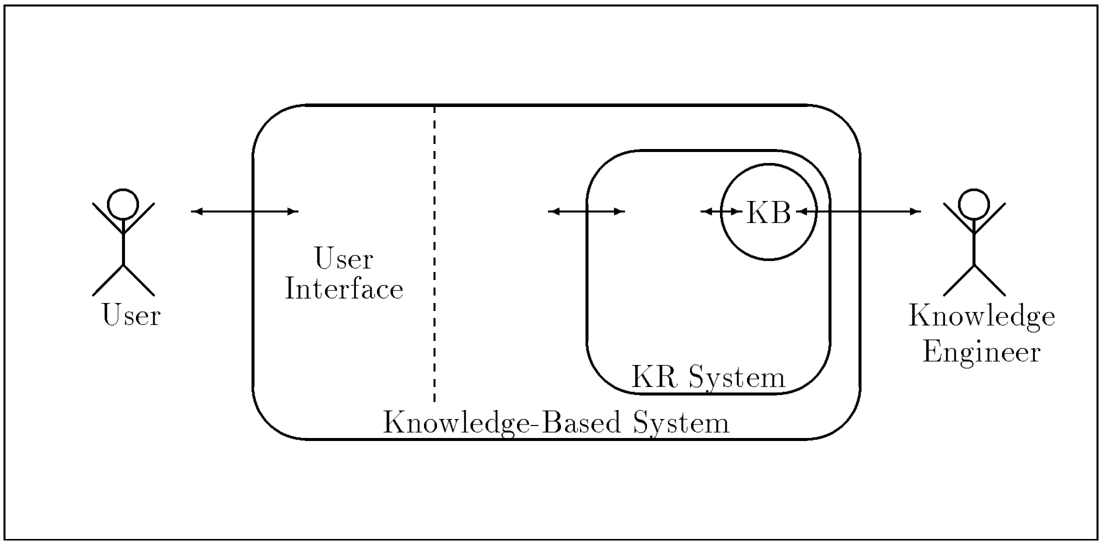

Figure 2.1: Architecture of Knowledge-Based Systems

If a system has to perform such complex behavior as participating in a natural language dialog, a couple of capabilities are required. For instance, it is necessary

[18]Of course, knowledge engineer and user can be the same person.

[19]This may result in embedding the knowledge-based system in the knowledge acquisition system, as it is the case in the TEIRESIAS system [Davis, 1982, p. 233].

to parse the input, to determine the meaning, to reason about the intentions of the speaker, to determine an answer, and to verbalize this answer. Evidently, the types of knowledge required for these tasks are quite diverse; actually, different domains are modeled (syntax, morphology, world-knowledge, knowledge about the partner, etc.), and the bodies of knowledge are only loosely coupled. For this reason, it makes sense to partition the knowledge into different sub-KBs which are usually called *knowledge sources*. Moreover, due to the differing requirements for different knowledge sources, diverse KR formalisms are usually used [Hoeppner *et al.*, 1983] – resulting in a *heterogeneous* KR formalism. This is quite different from the notion of a hybrid KR formalism as introduced above. In the case of hybrid representation formalism we require a strict connection between the subformalisms by insisting on a common semantics, while in the case of heterogeneous representation formalisms, this is not always intended.

Although these organizational issues are not trivial and many problems have to be solved in this context [Bergmann and Paeseler, 1985], we will abstract away them here, i.e., we will assume a monolithic KR system and KB as depicted in Fig. 2.1 – freeing us to analyze the problems of knowledge representation, knowledge management, and computational properties of KR systems on a more general level.

2.3.2 Services of Knowledge Representation Systems

One task a KR system has to fulfill is the transformation of symbolic expressions stored in a KB into a body of knowledge. However, a KR system is more than a static deductive calculus. A KR system has to provide services which allow accessing and maintaining the represented knowledge. This includes at least that the system is able to "assimilate new information" [Levesque and Brachman, 1985, p. 47].

Evidently, this is not enough. The world we are caught in is full of surprises, and it may turn out for a knowledge representation system that some things are different from what it believed. Usually the term *knowledge* refers to *correct belief*; therefore, by definition, a system representing knowledge cannot be wrong. Unfortunately, in reality any knowledge representation system can hold a wrong belief because somebody did not tell the truth about the state of the world, because some sensory data was inaccurate, or because something has changed in the world.[20] Sometimes there may be a more subtle reason for believing false propositions: it may be necessary to start off with some assumptions only to later discover that some of them were wrong.

Thus, besides drawing inferences and putting new knowledge into the KB, it is necessary to *modify* the KB. This is most easily achieved by directly manipulating

[20]For this reason, *belief representation* would be perhaps more appropriate, but I will follow the standard AI usage.

the data structures which are used to implement the KR system; a solution used, for instance, in the frame systems FRL [Roberts and Goldstein, 1977] and KRL. There are some drawbacks to the easy way, though. Arbitrary modifications of the data structures may blow up the entire system in the worst case, and in the best case, it may lead to results which have nothing to do with the semantics of the representation formalism – and then we might ask what all the fuss about semantics is good for. For this reason, it is often argued that a KR system should encapsulate the concrete implementational structures and provide only operations to the outside world which are related to the semantics of the KR formalism [Patel-Schneider, 1984; Brachman *et al.*, 1983]). In fact, a *functional approach*, hiding the implementation and focussing on the functionality, seems to be the most preferable solution [Brachman *et al.*, 1985, p. 532]:

> The user should know, at some level not dependent on implementation details, what questions the system is capable of answering, and what operations are permitted that allow new information to be provided to it. He needs to know how questions put to a knowledge base will be answered strictly as a function of the information it contains, and not dependent on how information is represented.

Thus, what is needed to manipulate a KB is at least a set of operations such as TELL, ASK, FORGET. In order to start off with something definite, we need, furthermore, an operation like INITKB, which yields an empty knowledge base. The important point here is that the operations should be related to the semantics of the KR formalism in a principled way.

As a matter of fact, as shown in [Levesque, 1984a] and [Brachman *et al.*, 1985], this can be done quite elegantly on the knowledge level for most of these operations. However, FORGET presents severe conceptual problems; problems we will return to in Sect. 2.3.4.

2.3.3 Inferential Services

From a computational perspective, the capability of turning a KB into a body of knowledge is the most interesting – and the most difficult to accomplish – feature of a KR system. Given a KR formalism with an appropriate semantics and a KB containing expressions of the KR formalism, it is clear what the KB claims to be true, namely, any proposition stored in the KB. For any nontrivial KR formalism, however, there are significantly more propositions entailed than explicitly stored. For example, in the terminology given in Fig. 1.1 the concept Modern-team has been defined without reference to the concept Small-team. Nevertheless, it is obvious that the former is a specialization of the latter (see Fig. 1.2).

Precisely this task, figuring out what has been specified only implicitly, can be said to form the heart of a KR system: to perform *inferences*. In the general case, this is nontrivial, often even noncomputable. Despite this fact, it is still useful to stick to model-theoretic semantics. Even if it is possible to only partially solve the problem, at least we know what is going on: it is clear what is claimed to be true, and it is possible to characterize the *inference algorithms* supposed to compute propositions entailed by a KB according to *soundness* and *completeness*, i.e., to ensure that the inference algorithms compute only entailed propositions, and check whether an algorithm is able to compute all entailed propositions, respectively [McDermott, 1978].

If we were interested in first-order predicate logic, *resolution* [Robinson, 1965] would be the technique employed to compute propositions entailed by a KB.[21] Algorithms based on this technique are sound and complete. However, the problem of deciding whether a given proposition is true or false according to a KB is only semi-decidable. This means we only get an answer for all propositions which are entailed by a KB. Propositions unrelated to a KB, however, can result in infinite computations.

For a subsystem which is supposed to provide services to another system, it is unsatisfactory that it is impossible to predict whether the subsystem can always answer a given question in finite time. A way out could be to restrict the expressiveness – the epistemological adequacy – of the KR formalism. For instance, if only monadic predicates were allowed, the resulting calculus would be decidable. Thus, there is an obvious tradeoff between expressiveness and computational costs [Habel, 1983].

Decidability is, of course, a desirable computational property, but it is not enough. It means only that a given problem can be decided in finite time by an algorithm. This can still be more time than anyone is willing to spend waiting for an answer, say a day, a year, or longer than our sun is expected to exist. And if in the worst case such computations are exponentially time bounded, a relatively small input can easily prohibit an answer in reasonable time. A more rigid and practical measure is computational tractability [Garey and Johnson, 1979], i.e., whether it is possible to decide a given problem in time bounded polynomially by the length of the problem description.

Usually, in AI, computational complexity is not considered to be the borderline where research stops. On the contrary, it is the line where interesting research begins. E. Rich [1983, p. 104], for instance, describes AI as "an attempt to solve NP-complete problems in polynomial time,"[22] and, if so, why should we be bothered by computational complexity? Rich's characterization of AI is at least partially accurate, but the key to "solving" the hard problems is to accept approximations as solutions and, more importantly, to employ *heuristics*.

[21]However, we will not pursue this line here. Readers interested in resolution techniques should perhaps try [Genesereth and Nilsson, 1987], which is a very readable and complete introduction to the use of logic in AI and to associated implementation techniques.

[22]Of course, Rich means intractable problems in general, not just the NP-complete ones. For instance, playing board games, which is subject to AI research, is mostly PSPACE-hard.

Thus, if one utilized knowledge in order to solve the hard problem of applying knowledge in order to solve the hard problems, then, at least, the application of *meta-knowledge* (or meta^n-knowledge) should be easy without requiring further knowledge. Otherwise there would be an infinite regress of heuristic knowledge. Moreover, there are a number of knowledge types where application of meta-knowledge does not seem to be possible – as argued in the eloquent article of Levesque [1986] about this and related issues.

This means that computational complexity is an important issue in knowledge representation. Unfortunately, however, even very restricted formalisms, such as propositional logic, are already computationally intractable. The problem of deciding whether a given formula in propositional logic is *satisfiable*, i.e., whether it is not always false, is *NP-complete*.[23] Two ways of dealing with this problem have been proposed by Levesque and Brachman [1987]. First, one could restrict the expressiveness of a given representation language, as exercised in that article. Second, one could accept approximations as solutions, e.g., by limiting the inferential power. In order to do this in a principled way, this limitation might be justified by an alternative, weaker semantics.

Levesque [1984b] and Patel-Schneider [1987b] pursued this second approach by employing variants of *four-valued logics* [Belnap, 1977]. Although this seems to be very promising in computational terms, a semantics for first-order predicate logic which makes entailment decidable and (almost) tractable is also very weak and thus does not support the obvious inferences one would like to draw (e.g. there are neither tautologies nor inconsistencies, and *modus ponens* is not a valid inference). This means that despite the fact that such a semantics gives a principled (model-theoretic) account to incompleteness, it is nevertheless not very useful because it is rather unintuitive. Moreover, in order to accomplish anything reasonable (pun intended), domain-specific inferences or a deductively stronger system must be used.

The main idea behind both approaches is "forging a powerful system out of tractable parts" [Levesque and Brachman, 1987, p. 89] because "as responsible computer scientists, we should not be providing a general inferential service if all we can say about it is that by and large it will work satisfactorily" [Levesque and Brachman, 1987]. In how far this goal is achievable, and what subtleties are involved in designing efficient inference algorithms will become more vivid in Chap. 4, where these issues will be discussed in the context of a concrete KR formalism.

[23] This means that the problem can be solved by a polynomial time algorithm on a nondeterministic Turing Machine. However, no polynomial time algorithm is known for a realistic processor, and almost nobody believes that there is one.

2.3.4 Revision Services

Inferring entailed propositions is the task necessary to support ASK operations. However, what is necessary to support TELL and FORGET? Are they supposed to work on the knowledge level (i.e., on all propositions which are entailed) or simply on the set of expressions we put into the KB?

In Sect. 2.3.2, we argued in favor of the functional approach – the user should know how questions will be answered as a *function* of the information a KB contains and independently of its implementation. The easiest solution satisfying this requirement would be to realize TELL as a "store" operation and FORGET as a "delete" operation – working on the set of expressions in the KB. Both operations are well-defined and the user can predict the response of an ASK operation without being forced to know anything about the implementation. From a computational point of view, there exists only one problem. If the KR system performs *assert-time inferences*, i.e., computes and stores inferred expressions when a new expression is entered into the KB, it has to check which of the previously inferred expressions are still valid after a FORGET operation.

While this seems to be a reasonable solution, the functional approach as described in [Levesque, 1984a] and [Brachman *et al.*, 1985] aims for more. On carefully rereading the quotation in Sect. 2.3.2, one will discover that the result of KB operations should be a function of the information the KB contains, *not dependent on how information is represented.* In other words, the KB operations should operate on the knowledge level [Levesque and Brachman, 1986] – not on the symbol level as the solution sketched above.

A formalization of the knowledge level view could be, assuming a model-theoretic semantics, the set of all models [Levesque, 1984a]. An alternative formalization could be to use the set of all true propositions [Diettrich, 1986]. Of course, this is not an implementational model, but serves merely as an elegant abstraction describing the competence of a KR system. If we are dealing with a monotonic formalism and are only interested in TELL and ASK, everything works out fine. However, if we are interested in nonmonotonic formalisms, or if we wanted the KB to forget something, we are in trouble.[24]

The task of eliminating a theorem from an arbitrary logical theory is underconstrained, i.e., there is no unique solution. And although it is possible to analyze the nature of such operations [Alchourrón *et al.*, 1985], it does not help us to derive a general specification for a FORGET operation in arbitrary logical theories – pragmatics have to be taken into account. This can include minimality criteria, the syntactic form of the stored propositions, the probable intentions, etc. In particular, if we have a restricted formalism and if we are able to pin down the intentions of a FORGET operation, we may be able to give an unambiguous, intuitively plausible specification of such an operation. However, we will most

[24]See also Ullman's comment [Brodie and Mylopoulos, 1986b, p. 57] on the knowledge level view.

likely have to take into account the symbol level.

Moreover, often enough it seems appropriate to modify not the consequences of a set of expressions, but the expressions which led to the consequences in order to realize a FORGET operation. If we think, for example, of a legal code, it seems intuitively plausible to modify the code and not the consequences [Makinson, 1985], and the same holds for other applications as well [Ginsberg, 1986].

Thus, the claim made above that all operations should be performed only on the knowledge level seems to be too strong. In fact, Newell intends to describe only the potential for actions, not how knowledge gets manipulated. The notion of knowledge as viewed by Newell and used in the functional approach is a static and unstructured object [Newell, 1982, p. 101]:

> However, in terms of structure, a body of knowledge is extremely simple compared to a memory defined at lower computer system levels. There are no structural constraints to the knowledge in a body, either in capacity (i.e., the amount of knowledge) or in how the knowledge is held in the body. Indeed, there is no notion of how knowledge is held (*encoding* is a notion of the symbol level, not knowledge level). Also, there are not well-defined structural properties associated with access and augmentation. Thus, it seems preferable to avoid calling the body of knowledge a memory.

If we consider *actions on knowledge*, the abstraction is too coarse, as we have seen. On the other hand, a pure symbol level account as proposed first also has its deficiencies. Assume we told a KB that $(a \wedge b)$ is true, and, later on, try to convince the KB that it should forget about the truth of a. Shall we ignore this request, or shall the proposition $(a \wedge b)$ be deleted?

Another point where the pure symbol level view seems inappropriate is that a "symbol level"-TELL operation is insensitive to its result. It does not matter whether a TELL operation leads to an inconsistent state of the KB. However, what one would like to have is a consistency-preserving operation which perhaps first changes the KB so that the new expression fits in. These problems, and a few more, will be the subject of Chap. 6.

2.4 Knowledge Base Management Systems

Some of the topics treated in this chapter may provoke the question whether another field in computer science – data-base research – could contribute insights and solutions to the problems described – or *vice versa*. As a matter of fact, a quite fruitful exchange of ideas between both fields has been taking place since approximately 1980 [Brodie and Zilles, 1980; Brodie *et al.*, 1984; Brodie and Mylopoulos, 1986b].

The interest of data-base researchers in AI is driven mostly by the feeling that conventional data base management systems (DBMS) are not expressive enough and that they do not support the kind of processing one would like to have in order to support the user. In short, data base management systems should become more "intelligent." Conversely, AI researchers are interested in "scaling up" KBs, in overcoming their space limitations by adapting techniques developed in the data-base field in order to manage large amounts of knowledge. The common vision is the development of a new generation of software systems, *Knowledge Base Management Systems* (KBMS), which are similar to DBMS, but manage knowledge instead of data.

However, though the problems being worked on in both fields may seem similar at first sight – something is stored, accessed, and manipulated in a data base or knowledge base – there are fundamental differences concerning requirements, underlying assumptions, and expectations. Brodie and Mylopoulos [1986a] summarized these difference in the following way:

> Any knowledge representation language must be provided with a (rich) semantic theory for relating an information base to its subject matter, while a data model requires an (effective) computational theory for realizing information bases on physical machines.

Of course, these are not inconsistent, but only diverging goals. There does not seem to be a genuine problem in envisioning a system which meets both goals. And in fact, when implementing a KR system, one tries, of course, to realize an efficient system. The measures of success and the underlying assumptions are different, though.

For instance, since the amounts of data stored in a data base usually do not fit into main memory, it is folklore in data-base research that the data has to be stored on secondary storage media. Therefore, most algorithms and techniques were developed with this assumption in mind. On the other hand, structures stored in knowledge bases are usually assumed to be stored in the main memory because otherwise the AI system would be too slow. Thus, approaches to utilize a conventional DBMS as storage machinery for a knowledge base in order to "scale up" cannot solve the problem by mapping one data-structure onto another one (as in [Härder *et al.*, 1987]) because both requirements – speed and size – have to be met, which requires a new generation of computers as well as new data-management techniques and algorithms.[25]

Besides the management of large amounts of data, there are other advantages one gains by using a DBMS (sharing, security, recoverability, etc.). Therefore, one might argue that even though data base management systems cannot be utilized

[25]Moreover, as Wiederhold et al. [1987, p. 378] argue, there does not seem to be an urgent need to pursue this line of research. He believes that "for all large AI systems the amount of complex stored knowledge, by our definition, is quite modest . . . ."

directly in KR systems, at least some important techniques may be borrowed and combined with KR technology in order to realize a KBMS. Again, one should be aware of the fact that the underlying assumptions about the application environment are fundamentally different. What is judged to be important for a data base application may be superfluous or even a handicap in an AI application [Bobrow, 1986].

In order to make this statement more vivid, let us envision a KBMS managing concepts, assertions (as sketched in the Introduction), rules, and perhaps other kinds of KR formalisms. Furthermore, let us assume that we have at hand all the operations and techniques we usually get with a full-fledged DBMS (security, concurrent access, transactions etc.). Such a system could, for example, be utilized in the development of a knowledge-based system: the developers can share the KB. However, if one member of the team changes, say, a concept definition, the effect of this change has to be propagated through the entire KB (which may take some time), and all other members of the team are suddenly faced with a new concept definition which may change the behavior of the knowledge-based system in unexpected ways. The problem is obvious: in a KB, the stored information is usually highly interconnected, and, moreover, programs may depend on a certain contents of the KB.[26] In order to avoid this problem, one could restrict updates in such a way that only one person can change the KB. However, this would limit the flexibility considerably. Thus, another solution is necessary here. For instance, the versioning method used in the LOOPS system [Bobrow and Stefik, 1981], which permits *private* updates, seems to be more appropriate.

What I am saying here is not that any exchange of ideas and techniques between data-base research and AI is doomed to failure. However, a naive approach in trying to transfer techniques based on implicit environmental and implementational assumptions or using existing software products, will probably result in inappropriate solutions. There is no ready-made technology just sitting around waiting to be integrated into a new generation of software systems. Instead, the pragmatic foundations of the techniques have to be made explicit, and implementation techniques have to be tuned to support them, perhaps using open and extensible system architectures as argued in [Carey and DeWitt, 1986].

If we set aside the ideas of implementing a KBMS in the near future and of "solving" KB problems with a DBMS, however, there are a number of important issues – practical and theoretical – which can be, or have been already, subject to research efforts in both fields.

More practical oriented approaches include

- the "enrichment" of the data-base query interface by a knowledge-based system, for instance, a natural language dialog system [Nebel and Marburger,

[26] Actually, in a data base context, a change of the data base schema, which can be seen as similar to the change of a concept definition, can, for good reasons, only be performed by a central institution.

1982; Marburger and Nebel, 1983], which simplifies the interaction. As a matter of fact, for simple data bases, you can already buy such an interface for your personal computer [Hendrix, 1986];

- employment of a DBMS as one component of a KR system in order to store simply-structured facts as described in [Wiederhold *et al.*, 1987; Abarbanel and Williams, 1987];
- using ideas from KR in order to develop new data models [Mylopoulos *et al.*, 1980];
- using KR systems to support query formulation [Tou *et al.*, 1982; Patel-Schneider *et al.*, 1984].

From a theoretical point of view, the semantics of data models and KR formalisms, as well as the semantics of operations on data or knowledge, are a field on which the approaches are converging and fertilizing each other. Although Brodie and Mylopoulos [1986a] make a distinction between knowledge bases and data bases along the line that the former is more concerned with (external) semantics and the latter is more concerned with efficiency, efficiency is, of course, an issue in KR (a point we discussed in Sect. 2.3.3), and semantics of data models is subject to theoretical analysis in the data-base field as well [Reiter, 1984].

Brachman and Levesque [1986] propose to analyze both kinds of systems under a unifying view, which should be the knowledge level view, abstracting from implementational issues (the symbol level) and from the system engineering level (presentation and organization). The idea behind this proposal is that if we gain an understanding of a system on the knowledge level, we actually know what is going to be *represented* by the system. As was pointed out in Sect. 2.3.4, this view is too constrained if we want to do more than TELLing and ASKing. It is still useful to stick to this notion, however, because it provides a unifying view on both fields and a yardstick for describing systems, even if they depart from the knowledge level [Diettrich, 1986].

The research described here could be understood precisely in this way. The knowledge level properties of a hypothetical system are analyzed, and the computational properties are derived from this view. In addition, problems which cannot be solved solely on the knowledge level will be analyzed on this level and the solutions will be related to it.