Psyke & Logos, 1988, *9*, 398-418

EXPERT SYSTEMS! WHO KNOWS?

Flemming Vestergaard

It is the purpose of this article to discuss in very simple, perhaps even simplistic, terms some issues related to the representation of knowledge in so called expert systems. As is well known the AI (Artificial Intelligence) business is haunted by often quite unrealistic claims and visions of possible applications; practitioners as well as non-practitioners contribute to this. This is one good reason to stick to simple terms and questions when discussing a subject as »knowledge representation in expert systems«. Therefore, I will not discuss, say, theoretical issues in frame-like representations, neither will I discuss whether it is inferencing techniques or knowledge representation techniques that may eventually fulfil some of AI's promises (cf. Lenat & Feigenbaum, 1987).

The subjects here are the embodiment of knowledge in expert systems, with a main focus on the »explicitness« of this embodiment and on the articulation of the knowledge in expert systems. Both issues are illustrated with simple examples. Instead of discussing subtleties, I will aim at illustrating what expert systems are all about in a somewhat more practical context. There will be a claim, however, that expert systems may indeed embody knowledge in ways markedly different from those of conventional systems (taken to be anything but AI systems) and that these ways of embodying knowledge may share important characteristics with the ways knowledge is represented in the human cognitive system. But note that there is nothing dramatic in this statement, it implies neither that machines may be like humans nor that humans may be machines (and by the way, I will not discuss topics in that vein).

There are six sections. As an introduction there is a brief discussion of what is understood by »expert system« in the present context, and rule based systems are illustrated with a detailed example of an extremely simple system. Next, I take a look at a few problems in more complex systems where rule based representations do not suffice. Then some concepts (an excerpt from a larger set) useful when describing knowledge are introduced, and the concept of domains of knowledge is discussed in terms of the psychology of memory. Finally, some concluding remarks.

1. Expert Systems

There are two main approaches to the definition of expert systems (hereafter ES). One identifies, essentially, ESs by the tasks they perform: ESs are systems performing tasks requiring, in pre-system time, a human expert or expert advising. It is an obvious shortcoming of this approach that a computer guide to an amateur astronomer's Schmidt-Cassegrain telescope could reasonably be considered an ES, as could desk top publishing systems, et cetera. I will stick to the other main approach which is more concerned with how the system works. Generally, an ES has a knowledge base consisting of rules and facts and an inference engine which matches rules against facts and, when matches are found, fires the rules, i.e., performs the actions stipulated in the rules, such as asserting new facts that may make other rules match the updated fact base. Rules have the form »IF <something> THEN <something>«, and a rule whose IF- or condition-part matches the fact base is said to be applicable. At any time a number of rules may be applicable, this set of applicable rules is often called the conflict set, and the inference engine must select one to be applied (fired) from this set (according to a so-called conflict resolution strategy which may be hardwired or programmable or even rule based). In this view an ES is characterized by the way it performs certain tasks - not by the nature of these tasks - and by the way knowledge is represented.[1]

Consider the following fragment of a potentially useful system:

Rule-836

IF there is no coffee
and someone wants coffee

THEN get some coffee

This depicts the general form of a rule: It has an if- or conditions-part and a then- or action-part; these will in what follows be called the left-hand-side (LHS) and right-hand-side (RHS) respectively.

Say, that the current state of the system containing rule-836 is such that the following facts are given (possibly among a number of other facts):

there is no coffee

kurt wants some coffee

In this case rule-836 is applicable, and if applied (fired) it asserts the new fact

get some coffee

which in turn may make one or more other rules applicable. Residing in a robot this intelligent system would, supposedly, trigger a coffee making behavior, perhaps implemented by means of a set of rules. Depending on the rule set and the current situation (i.e., the current contents of the fact base) this behavior may involve calling the plumber to have water installed, to buy some coffee, or simply to fetch the coffee pot in the kitchen. Or, the coffee-making behavior may not be triggered at all if, for instance, some other behavior is also called for and it has, say, a higher priority than coffee making.

Consider now a typical example of a LISP function definition:

```
(defun recursive-factorial (n)
   (if
      (=n 0)
      1
      (* n (recursive-factorial (1- n)))))
```

LISP novices are generally assumed to define this function which quite elegantly implements the common definition of n!. (Non-integer and negative arguments are not handled properly, however). Providing large arguments to the above function may cause problems due to stack overflow, and one may be tempted to define the function differently, for example as follows:

```
(defun iterative-factorial (n)
   (do
      ((x 1 (1+ x))
        (f 1 (* f x)))
      ((> x n) f))) ; f is returned
```

In a sense the two definitions do the same job. It could also be argued, and reasonably so, that the latter definition is somewhat more obscure than the former one, or otherwise stated that the former definition is the more explicit one, as compared to the text book definition of n!. This touches upon one of the main virtues often stressed with respect to ESs, namely that they represent or embody knowledge in a more explicit way than do conventional systems. An example would be the above rule-836 which can be considered a fairly straightforward and quite explicit representation of a piece of useful knowledge.

In the following section rule based systems are illustrated with a small ES where all relevant knowledge in the system is represented quite explicitly in reasonably modular chunks.

2. A Simple Rule Based System

Below rule based systems are illustrated with a small, but rather elegant ES. The system is a simple forward chaining system (i.e., a system that derives new facts on the basis of known facts); the initial facts known to the system are simple facts like »sigmund is father to anna« and the rules can all be considered straightforward definitions of kinship terms.

An example of a rule:

Grand-parent rule

IF ?parent is parent to ?child
and ?grand-parent is parent to ?parent

THEN ?grand-parent is grand-parent to ?child

Words with a leading ? are variables here. ?child and ?parent, for instance, will be matched against facts known to the system, and if the fact base contains, say

ed is parent to alan

?parent will be bound to »ed« and ?child to »alan«, and the first part of the condition of the rule is satisfied. Note that the second part of the LHS also matches the fact, but that the LHS as a whole is not satisfied. If the fact base also contains

douglas is parent to ed

the LHS as a whole is satisfied, and the rule asserts

douglas is grand-parent to alan

Simple expert system shells exist that do not allow variable in rules in this sense. Such systems should not be considered in the context of serious applictions; perhaps they should not even be considered for simple or toy ESs, the following example (which is extremely simple) would be awkward to implement in such a shell. The system expands on the above rule.[2]

The definition of the ES itself:

(defes kinship)

The initial facts have the form (father karl jenny), an arbitrary choice:

```
(deffacts kinship)
    (mother tytte claus)
    (mother tytte morten)
    (father flemming claus)
    (father flemming morten)

    (father tage flemming)
    (mother carla flemming)
    (father tage lisbeth)
    (mother carla lisbeth)

    (father jørgen tytte)
    (mother inger tytte)
    (father jørgen bitte)
    (mother inger bitte)
    (father jørgen gorm)
    (mother jørgen gorm)

    (mother bitte susanne)
    (father peter susanne))
```

Two rules that assert facts about the sex of parents:

```
(defrule
    fathers-are-male
    (father ?f ??)
    (male ?f)
    :in kinship)

(defrule
    mothers-are-female
    (mother ?m ??)
    (female ?m)
    :in kinship)
```

A rule which may be seen as implementing the concept of parents as a generalization of father and mother:

```
(defrule
    parent
    (or*
        (mother ?p ?c)
        (father ?p ?c))
    (parent ?p ?c)
    :in kinship)
```

A (full) sister is (a) female, (b) has a father, (c) has a mother, (d) her father is father to someone else, and (e) her mother is mother to this same someone. Likewise for brothers.

```
(defrule
   sister
   (and*
      (female ?s)
      (father ?f ?s)
      (mother  ?m ?s)
      (father ?f ?x ~ ?s)
      (mother ?m ?x))
   (sister ?s ?x)
   :in kinship)

(defrule
   brother
   (and*
      (male ?b)
      (father ?f ?b)
      (mother ?m ?b)
      (mother ?m ?x ~ ?b)
      (father ?f ?x))
   (brother ?b ?x)
   :in kinship)
```

A grand-parent is a parent to a parent of a child. Note that it is useful to have the parent rule defined above. A grand-parent's sex is also of importance:

```
(defrule
   grand-parent
   (and*
      (parent ?p ?c)
      (parent ?gp ?p))
   (grand-parent ?gp ?c)
   :in kinship)

(defrule
   grand-mother
   (and*
      (grand-parent ?g ?c)
      (female ?gp))
   grand-mother ?gp ?c)
   :in kinship)
```

```
(defrule)
   grand-father
   (and*
      (grand-parent ?gp ?c)
      (male ?gp))
   (grand-father ?gp ?c)
   :in kinship)
```

Obviously this small system could be easily enhanced to handle a number of generations, to ask for the sex of those for whom it cannot be inferred, et cetera.

When this system - which is certainly a rule based system and which may be considered an expert system - is run it generates new facts on basis of the initial facts. For instance, it will be known that

(sister bitte tytte)
(grand-father tage claus)
(grand-father jørgen claus)

It will not be known that

(sister lisbeth flemming)

because there is no rule to infer the sex of childless persons and the sister rule relies on sex.

As defined here the rules of the kinship ES fires pell-mell, at random in the sense that no specific sequence is defined; all rules that can fire does so, i.e., the system runs exhaustively. Rule firings can be controlled, and a specific goal could be set for the ES, but it would only clutter things in the present context. The fact that this system works properly without any specific control was the motivation for calling it »elegant« above. The same rules could be used in a backward chaining system with a goal such as (brother claus ?m).

In the next section somewhat more complex systems are addressed. Illustrations are, therefore, not as detailed as above, and the concern is more with how knowledge is embodied in the systems.

3. Embodiment of Knowledge

Above I have touched upon the notion of embodiment of knowledge. In a rather straightforward sense knowledge is embodied in a host of man made objects. For instance, knowledge is embodied in complex mechanical devices like diesel engines and gyroscopes, and knowledge is also embodied in

simple devices such as hammers and bottle openers. In a certain sense it is the embodied knowledge that essentially yields the usefulness of the devices; often one can on the basis of a general understanding of the purpose of a device infer the purpose of specific details and in this sense gradually reveal the knowledge embodied (cf. Rubin, 1920). (This theme is heavily discussed in social phenomenology, and it is one aspect of this *sens lourd* (Merleau-Ponty, 1959/61, cf. Schütz, 1932, two recommendable books) or inert meaning pervading our world I am getting at with the notion of embodiment of knowledge, but it is out of the present scope to discuss how information and AI technology changes this important aspect of the social world.

If a hammer embodies knowledge it is obviously the more so for practically any computerized system, being a function definition as above or a telescope guide. Clearly, the knowledge embodied in a hammer does not preserve much - if anything - of the characteristics of »hammering knowledge« as represented in the human cognitive system, but it could be reasonably argued that computer systems are potentially closer to this and, perhaps, that ESs may preserve some essential characteristics of human knowledge.

Looking at the kinship ES described in the previous section, it has a feature making it markedly different from most programs: The knowledge embodied in it is quite explicitly represented; each rule is a definition of a kinship term - dependent on only a specific notation (the rule language) and the choice of fact format - and »porting« the ES to other cultures would be a simple matter of revising definitions. In this system the rules are, furthermore, modular »chunks« of knowledge, an often acclaimed virtue of rule based systems. More often than not it is not that easy to build even small rule based systems, however; rules tend to interact or trail each other in ways much more intricate than in this example.

Consider now some of what would be needed in order to enhance the knowledge base of a domestic robot able to make coffee properly. An ES at work here must have explicit knowledge about structure, for example:

A coffee maker comprises of

one heating element
one water reservoir
one water tube
one coffee pot
one heating plate
etc.

And the systemm would need rules such as

Excess water rule

IF water reservoir is not empty
and coffee pot is full
and flow in water tube > 0

THEN water will flood the kitchen

Additionally there would have to be some representation of the strategy to follow when making coffee. For instance, the robot should add coffee beans before turning on the machine, or when looking for faults it is appropriate to check the power supply before disassembling the machine. Concerning this latter aspect it is worth noting that ESs are potentially able to reason about their own knowledge: Due to the explicit representation of, e.g., rules a rule based system can reason about which rules to apply or not to apply in a given context (in principle, that is, some can, some cannot).

One way to organize issues as the above is to distinguish between three layers of knowledge representation in ESs:

MODEL LAYER:
Knowledge about the object of reasoning.
(For instance, the various parts of a coffee maker and their functional relations).

OPERATOR LAYER:
Knowledge about how to reason about the object, i.e., a representation of allowed or desired inferences with respect to the object.
(Cf. the excess water rule above).

STRATEGY LAYER:
Knowledge about problem solving or reasoning strategy.
(Generate and test one hypothesis at a time, or generate all hypotheses and select the n most promising, for example).

Generally, a distinction like the above can be drawn with respect to any ES. But often there is no similar distinction to be found in the ES's representation of knowledge. A rule based system solving the problem of the farmer, the wolf, the goat, and the cabbage typically does not have any explicit representation of, among other things, the boat's capacity; this crucial piece of information is hidden, as it were, in the formulation of one or more rules (much as the meaning of n! is hidden in a function definition). Similarly, the kinship ES does not have any representaiton of the fact that there could be no more than one (biological) mother for any person; were the system reasoning about marriages some representation of »max number of wives« would obviously be needed.

For a number of reasons a common format for the three layers are not de-

sirable in complex applications (in »classical« rule based systems only rule format is available, however). But the distinctions are not necessarily distinctions of form. The three following rule fragments illustrate this.

Adder-definition rule (Model Layer)

```
IF      input1 = x
   and  input2 = y

THEN    output = x + y
```

Defective-adder rule (Operator Layer)

```
IF      input1 = x
   and  input2 = y
   and  output = x * y

THEN    adder defect
```

Apply-proper-rule (Strategy Layer)

```
IF          known = input1, input2, output

    THEN    apply defective-adder
```

The first rule defines a property of certain objects, called adders, the system is supposed to reason about; the second rule identifies one (weird) way such an object may be defect; and the third rule defines when to try a specific rule. (The meaning of a rule like the third one is to make sure that a rule such as defective-adder is only tried if there is sufficient information about an adder to actually test the LHS of the rule. Rules like this can be used to control rule firings, or for performance reasons).

The operator and strategy layers are distinguished from each other because they refer to basically different kinds of knowledge. The operator layer embodies what I will call entity knowledge, i.e. knowledge about structure or behavior of entities in a domain. The defective-adder rule above states that an adder yielding the product of its inputs is defect. Similarly the adder-definition rule is a piece of entity knowledge. The model layer embodies knowledge about the object as something distinguished from and independent of the reasoning about the object, whereas the operator layer contains knowledge about how to reason (which inferences to make) with respect to a specific (kind of) object. The apply-proper-rule rule, on the other hand, embodies what will be called control knowledge. Control knowledge such as if an adder is to be tested, the first thing to check is if the input values

are known, and if this is the case they should then be fed to a comparison with the output value, otherwise some other action could be taken (such as asking for the values). (Cf. section 4 below).

Especially as regards the model layer rule based representations are not particularly adequate; it is also clear that representations in the form of simple facts will not do the job (for practical reasons, not because of any deficiency in principle). A number of ES building tools/systems support object oriented representations called, among other things, frames, classes, concepts and schemas (there are differences between these, but they will not matter here). These systems are frequently called hybrid systems because of the dual knowledge representation, rule based and object based.

It is common to hybrid systems that they allow definition of, say, the concept of a water reservoir for the coffee maker above. A water reservoir is defined, then, as a concept with attributed such as

maximum water level
current water level
temperature
etc.

Additionally it is possible to attach functionality to concepts by defining methods (a method is kind of an analogue to a procedure in a conventional programming language). And the systems support, furthermore, inheritance among concepts, i.e., the inheritance from superordinate concepts of attributes and methods; the inheritance supported may be single (linear) or multiple (non-linear). In this case the water reservoir might inherit from more general concepts - a concept of coffee maker parts, a general container concept, or something similar - and reservoirs for specific makes of coffee machines may inherit from the present concept. When an adequate set of related concepts are defined models of specific objects can be built by creating instances of the concepts.

It should be fairly obvious that ESs do often embody knowledge in a way that differs in important respects from the ways knowledge are embodied in mechanical devices and conventional computer systems. First, the knowledge embodied is so in an explicit way markedly different from function definitions and hammers. Second, this allows for explicit representations also of control knowledge, i.e., knowledge about other pieces of knowledge. And third, a distinction can be drawn in ESs between the stock of knowledge and the application of this knowledge to specific problems. This last distinction can also be drawn with respect to human problem solving.

It is not a problem to describe those aspects of a coffee maker relevant to finding simple faults in the device by means of facts, for example. But considering more complex objects - power plants, geological structures, orga-

nisms, et cetera - the model layer of ESs cannot be handled without more advanced means for representing structure etc.

Object oriented systems sometimes claim that they allow a direct mapping from natural, real-world concepts to classes/concepts. This is normally not the case (and never the case with systems without multiple inheritance), but they do support such a mapping to an extent not found elsewhere (or only in specialized applications), and it is clear that these forms of representation can allow for model layer knowledge representations that are as direct and explicit as are the rules of the kinship ES. Hybrid systems thus have a potential for representing and embodying knowledge in ways that are powerful as well as attractive when compared to human knowledge representation. The strategy layer is the most problematic one, since it is not at all clear whether or when rule based, object based (i.e., control models), or hybrid representations are most appropriate.

Summing up, I will say that ESs that rely on knowledge representations as described till this point do in fact embody knowledge in ways that are not matched by conventional systems (and not by hammers etc.). They do so not only because rule and object based representations are both powerful, and not only because they can be mixed - each for adequate purposes - in hybrid systems, but also because an ES has a potential for reasoning about how it represents knowledge itself. In the next section some concepts about knowledge itself are introduced; they pertain to ES knowledge embodiment, but also to human knowledge representation.

4. Some Distinctions Concerning Knowledge

In this section a few concepts concerning knowledge are introduced in order to allow for a characterization of how ES knowledge embodiment may be in at least one important respect similar to human knowledge and different from the embodiment of knowledge in conventional systems.

Knowledge can be seen as an ensemble of knowledge stock and knowledge application. (Where I would say that problem solving is the application of knowledge to specific problems). An example in order to clarify what is meant by this distinction: A person possesses a stock comprising of the multiplication table from 2 to 10, and when multiplying 47 and 69 he applies this stock. However, the stock does not constitute all of his multiplication knowledge; it is not part of the multiplication table from 2 to 10, that it can be applied to to the problem »47 * 69 = ?« or »78654 * 2/5 = ?«.

The stock provides the context for interpretation of problems; this implies that problem solvers may interpret identical problems differently. This is quite obviously so in the context of human cognition, but it is also clear, from the simple illustrations given in previous sections, that ESs also may interpret problems differently. The stock determines what the knowledge can be

about; it determines how the problem is actually interpreted: Having defined the concept of a pump, pumps may be represented, and the definition of the concept, such as the attributed defined for pumps, determines how a specific pump can be interpreted, e.g. if the number of revolutions per minute can be known or not. The application of knowledge, on the other hand, determines what the knowledge can be used for: Quite simply, number of revolutions per minute may be known for pumps, but if there is no rule referring to this feature, this part of the knowledge stock will not be applied to any problem solving.

With respect to knowledge stock as well as application it is possible to distinguish between object and form. The object of the stock, then, is seen as the ensemble of entities that can be represented in the stock of knowledge, and the object of the knowledge application is the problem as interpreted in the context of the knowledge stock. The form of the stock concerns the type of relations realized in the stock, and the form of the application of knowledge is the manner of applying the stock of knowledge to the problem.

An example: I for my part has a stock of the alphabet in sequence,

a, b, . . ., y, z.

In order to get the relative position of, say, »m« in this sequence I have to unravel the subsequence »k, l, m, . . .«, while any librarian is able to access the relative position directly. That is, while the librarian and I may rightly be said to possess the same stock of knowledge concerning the alphabet, the form of the stock varies, causing considerable differences in problem solving behavior (and, in this case, problem solving performance). The object of knowledge application is the problem as interpreted in the context of the knowledge stock. The object varies with the stock, then, and for instance the object of knowledge application is not the same for the librarian and I, even if the problem (like the present one) is easily identified out of context, as it were. Quite similarly the possibilities of accessing the information stored in the multiplication table influence application via the interpretation of the problem.

The form of knowledge application is the manner of applying the stock to the problem. What is hinted at here can, again, be illustrated with the alphabet problem. Faced with a number of subproblems such as finding the relative positions of »m«, »n« and »o«, I may apply my (somewhat deficient) stock of knowledge like this:

»k, l, M, . . .« «k, l, m, N, . . .« »k, l, m, n, O, . . .«

Or I may be as smart as to note that »m, n, o« is a subsequence of the alphabet allowing for a more efficient solution to the problem. In contrast the librarian may simply access the relative positions of each letter directly, per-

haps without noting that they form a subsequence. In the context of ESs concepts such as »reasoning mode« and »control of knowledge« often refer to the form of knowledge application.

In the previous sections I have talked about entity and control knowledge. By entity knowledge is simply meant knowledge whose object is the entities in a certain area. Whereas control knowledge (sometimes referred to as »meta-knowledge«) is knowledge whose object is the form of stock and application within an area. Control knowledge is defined as knowledge about the form of knowledge, that is, control knowledge is knowledge about how entities are represented in the knowledge stock and/or about how the stock is applied to problems; note that control knowledge is not only knowledge about knowledge application but also knowledge about the stock (this is not commonplace, but it is naturally when the knowledge stock, as it is here, is seen as heavily influencing problem solving because the problem to which knowledge is applied is interpreted in the context of the stock). Note that we are talking about control knowledge, not about control reasoning; control knowledge, like any other knowledge, comprises of a stock as welll as knowledge application. Control knowledge is simply some other knowledge applicable to a given problem; it is not something very special. And note finally, that no hindrance is stipulated for having knowledge about knowledge about the form of some entity knowledge, i.e. control konwledge about control knowledge.

5. Domains

Often the set of problems to which an ES can be applied, i.e., the areas where it is knowledgeable or manifests expertise, is called a domain (and what has here been called entity knowledge is often called domain knowledge, then). Similarly, one can talk about domains of knowledge or expertise with respect to human experts, and one often does. In this setting it is rather natural to investigate how human and machine domains may map into each other. This is not a simple thing, however, since one must both have a general concept of domains and an understanding of how they may be organized in machines as well as humans. Below a few issues relevant in this context are discussed. First some observations about domains in general.

Quite commonly a domain is characterized by its object, i.e., what the domain is about. This is probably adequate in many contexts; this or that ES or person knows a lot about gas fired boilers but only little or nothing about chess and vegetables. However, there is a pitfall here, since domains cannot possibly find their definition in sets of real world entities - and this is sometimes what is actually implied when a domain is characterized by what it is about. Domains of knowledge reside within the realm of cognition (human or machine) and the definition of what is a domain must also be in the scope

of cognition. Certainly, knowledge is knowledge about something, knowledge has an object, but this does not imply that a domain of knowledge can be identified by pointing to the non-cognitive entities encompassed by the domain. The chef's expert knowledge cannot be defined by enumerating vegetables, fish, dishes, et cetera. Tomatoes are neither kinds nor pieces of knowledge.

Take another look at this. Let D be a domain defined by a set of entities, D1 be P1's knowledge about D, and D2 P2's knowledge about D. We would say - were domains defined by non-cognitive entities - that P1 as well as P2 has knowledge about D, provided that neither D1 nor D2 is empty. This may indeed seem quite reasonable. But note that we would have to assume this quite irrespective of how P1 and P2 contualizes D. The consequence of this is rather absurd, for instance a botanist's and a chef's knowledge about vegetables would be made equivalent in quite a misleading way.

Furthermore, domains are - as they can be interpreted in the context of human cognition - not static structures or organizations of knowledge. Organizations of knowledge are something realized in specific contexts, that is, domains should rather be understood as transient organizations of knowledge. One aspect of this is that knowledge common to several domains (i.e., subdomains), e.g., practically any person's knowledge about multiplication, should be seen as being part of a number of domains, and it is not naturally to conceive of this subdomain as being duplicated in a number of static structures. Rather, it is a domain which is sometimes realized as a subdomain of one or another domain, sometimes not.

In human cognition domains can be identified in several ways, e.g., on the basis of conceptual structure. I shall take here a more basic or elementary approach and spend a few words on the definition of domains on the basis of the psychology of memory.

It is natural, in my view, to think of domains of knowledge as sets of long-term memory (LTM) entries tending to be retrieved together. And the point, then, is to define what is meant by the phrase »tending to be retrieved together.«[3]

Memory is partitioned into unitized items. An item is an »object« or »chunk« in memory which may be sampled during a memory search. Items are the elements of memory, but they are not elementary in the sense of being simple and uncomplicated; an item may be a complex information structure and items may overlap. Retrieval from LTM is cue dependent and probabilistic. The basis of retrieval is a probe cue (or a set of cues) which determines what may be retrieved. A given set of cues has some chance of retrieving any item in LTM. Successive uses of the same probe may result in retrieval of different items.

Retrieval is based on the strengths of associative relationships between probe cues and memory items. These strengths are described in a retrieval structure, which is a matrix of retrieval strengths of each cue to each item.

The retrieval structure is dependent on the cue set; the strength between any given cue, Ci, in a cue set and a specifik item, Ii, varies with the cue set. During a memory search individual strengths in the retrieval structure are combined into a single activation for a given item. The implication of this is that multiple cues allow focussing of the memory search. Note that this is an important way for items to tend to be retrieved together.

There is a distinction between sampling of items and recovery of an item in memory retrieval. Sampling of an item means that it is activated above a specifik threshold; this activation is a condition for recovery, which means that the item is entered into short-term memory. Items may be sampled during a memory search, but only one of them recovered. There is a strong tendency that items are sampled from the most dense region of the intersection of the associative fields of separate cues. Or more simply stated, multiple cues increase sampling probability and may override a higher strength on one cue. Probability of recovery rises as cue-to-item strength rises. That is, the probability that an item which is sampled is also recovered increases if there is at least one very high strength (as compared with the other strengths in the matrix) relating a cue and the item in the retrieval structure. (These items are those that »come to mind« immediately, those that are »the first I think of when . . .«). The stengths relating recently sampled items to cues tend to rise, that is, recent sampling will increase probability of sampling as well as probability of recovery for the item in question. One important consequence of this tendency is that items related to a given context, such as a specific task, are likely to be retrieved in successive memory searches. Conversely a context shift, as imposed by, e.g., an external interruption of a task, decreases the probability of sampling and recovering items belonging to the original context.

A psychologically reasonable account of what it means to be »tending to be retrieved together«, on the basis of the above, used in a definition of a domain of knowledge results, then, in saying that a domain is a region of a retrieval structure dense with high strengths.

This can be expanded a little by taking what may be called inter-cue and inter-item connections into account. The retrieval structure determines relations between cues and items. An inter-item connection exists when one item has the ability to cue another item. Inter-cue connections are of a somewhat different nature. Cues are conneced when they are conditioned by (or, belongs to) a recurring situation or context. From the cognitive point of view inter-cue connections provide a link to the situation in which retrieval takes place, such as a job environment, a user interface, or whatever.

Given these concepts the core of a domain can be understood as a region, dense with high strengths, of an intersection between connected items and connected cues in a retrieval structure. (And what may be termed the fringe of a domain would be those parts of the domain that do not belong to the core). This definition of domains fits reasonably well with more common

sense understandings of a domain as a tightly knitted lump of knowledge. (But note that common sense would probably not see cues as co-determiners of the domain, and note that cues are only defined in the scope of retrieval, i.e., they are not something outside the realm of cognition).

Some consequences of conceiving domains of knowledge as here should be pointed out. First, borders of domains - their demarcations from other domains - are fuzzy. The main reason for this is that strengths are relative and retrieval probabilistic. An alternative way of looking upon this is to think of domain membership as a matter of degree. Second, strengths, in the retrieval structure do not alone determine domain membership. A high strength relating a cue and an item may fall outside a domain when no inter-item or inter-cue connections are present. (In both cases the item in question may be considered more loosely connected to the domain). And third, domains may overlap: That is, an item may be part of more than one domain and subdomains may be part of more than one superordinate domain. When domains overlap they are related to each other or connected; not only do they share items and cues, they may also, and for this reason, cue each other in a certain sense, i.e., being in one domain can facilitate access to related domains.

One main thing to observe is that the concept of a domain is somewhat undermined, as compared with the common understanding of domains in AI (cf. the beginning of this section). Even if domains are identified as here - as dense regions of retrieval structures - there is no real basis for identification and investigation of a singular domain. The reason is simply that domains of knowledge overlap and that what may be seen as a domain in one context is a subdomain in another. That is, knowledge is certainly organized, and organized in areas or domains, but they may all rightly be considered subdomains, in the sense that they are all inclined to be part of each other. To put this another way: An expert's knowledge can hardly be identified as one singular domain of knowledge, it is rather a composition of a number of domains; this composition, as well as each component, is transiently organized. Obviously, this state of affairs make it hard to provide a reasonable mapping from human domains of knowledge to machine implemented domains (as it is at issue in what is mostly called knowledge acquisition). An important aspect here is that the sharp distinction in any ES between what is known and what is not has no correspondence in human knowledge organization.

I shall not go into detail with how machine domains, such as the domains where ESs are expected to show expertise, differ in organization from human domains of knowledge as depicted here. There is a number of obvious differences. Perhaps the only obvious identity is that machine domains can also be conceived of as relying on cue dependent retrieval in most cases; in contrast retrieval is generally not probabilistic on machines, and probably no one would like it to be (outside the scope of simulation of human cog-

nition). (For another perspective, see Slatter, 1987).

This issue is of practical importance in industrial ES building. The major building systems provide means for defining universes of discourse (as they are often called) or domains. But no system allows an organization of domains which allows - to take what I think is the most important point here - creation of an organization reflecting the relations between sub- and super-domains identified above. Aspects of interaction with ESs are frequently discussed in the context of AI, and often the problems identified as well as the claims are rather unrealistic as seen in the context given here; but problems related to what has been discussed in this section do make it far beyond present day technology (at least in industrial settings) to consider ES-man interaction as something potentially similar to the interaction between an expert adviser and a client.

6. Concluding Remarks

ESs promise - it is often assumed - to accept simple and transparent declarations of knowledge and do something reasonable with them. Generally they do not keep this promise. This should be clear from the above discussions, even if they do not illustrate the problems in systems as complex as those to be considered in the context of serious applications.

It is also clear, however, that expert systems may indeed embody knowledge in forms that are more similar to knowledge as represented in the human cognitive system. Not that there has been any conclusive evidence for this in the present short introduction. The claim rests mainly on the following observations related to the above discussion.

Expert systems represent knowledge explicity. This form of representation is different from the way a desk top publishing system, for instance, represents or embodies its knowledge about publishing tasks. One consequence of this explicit representation is that expert systems are potentially capable to reason about their own knowledge; most systems do not make use of this capability, but a number of them have the potential. This potential is not found in conventional systems, and it is a potential shared with human cognition.

Another implication of the explicitness is that the so-called knowledge base of expert systems may be much more easily created and maintained than it is the case with conventional systems. One promise of expert systems is that those persons that actually possess the relevant knowledge enter the knowledge to the system themselves and in a form not too dissimilar to the form this knowledge has outside the expert system context - and even in a form more than remotely related to how the knowledge is represented in the human cognitive system. However, present day expert systems do not nor-

mally keep this promise. Rule bases are notoriously difficult to keep consistent and debug, for example, but in some respects there is some truth to the claim, especially in the context of object oriented representations as briefly described in section 3.

In section 4 some concepts related to the description or categorization of knowledge were introduced. There was a distinction between knowledge stock and knowledge application, and there was a distinction between object and form with respect to both. These distinctions can also be identified when the concern is with expert system representations of knowledge. This is basically why that knowledge may resemble human knowledge in some respects.

Probably the mose decisive aspect making expert systems different from conventional systems is that it is possible to distinguish the knowledge embodied in the system from the application of this knowledge to specific problems. In my view this mirrors an important aspect of human problem solving, and nothing similar is found in other systems.

But note also that the issues discussed in section 5 point to problems in knowledge organization where no »solution« is near. It seems to me that there are features of human knowledge organization that are far from being mirrored in expert system knowledge representaiton, and these features seem to be important for manifesting expert behavior in the human sense.

Finally, a few words of moderation. The preceding remarks should not be taken to mean that I share claims common in the AI business that ESs can be created by anyone knowledgeable in a domain, that they may outperform human experts in important aspects, and that they may solve problems hitherto intractable. I do not think that any of these claims are true. In particular, they are certainly not true when talking about present day industrial applications. I have pointed to some features of expert system that make them different from mechanical systems and conventional computer systems. But by and large it remains to be shown whether these differences make any important difference.

NOTER

1. There is no generally accepted definition of ESs; and there is no generally accepted definition of AI. Many systems called expert systems do not contain rules at all; later I will also take a short look at representations that are not rules (rather a more complex kind of facts), but in the context of hybrid systems that also have a rule base. This should not be taken to imply that I find that rules are in fact a decisive characteristic of systems that could be dubbed »expert«; but rules in some form are an important aspect of many (or even most) expert systems, and notably the »classics«. In this short introduction I consider only systems that are at least partly rule based, using the term »ES«. For instance, neural network systems may also rightly be considered expert systems, but they are beyond the scope.

2. In this example I will not use the arbitrary format for rules used hitherto. I will use the format of the ODIN/TOR system developed at Søren T. Lyngsø A/S, AI Division. The use of this formats signals that the definitions given make up an actually running system. Only the most basic part of the language is used, viz., »and*« meaning »and«, »or*« meaning (inclusive) »or«, and »not*« meaning »not«. The syntax of the forms used in this article is as follows:

   (defes <name of es>)

   (deffacts <name of es>
   <list of facts>)

   Facts are lists, e.g., (this or that) or (this (or (that)) (may happen)).

   (defrule <name of rule>
   <LHS>
   <RHS>
   :in <name of es>)

   LHSs as well as RHSs are also lists, i.e., lists of patterns to match against the fact base. For example,

```
(defrule
    rule1
    (and*
        (or*
            (a b)
            (and*
                (c d)
                (there is no ?c)))
        (someone wants ?c))
    (get some ?c)
    :in my-es)
```

   is a valid rule.

   Additionally »~« is used as a kind of negation: In a rule

   (<variable> ~ <variable or constant>)

   means that <variable> should not be bound to the value of <variable or constant>.

   ODIN/TOR (and other large shells or expert system building systems) provide languages much more powerful than what is depicted here, they allow for explicit control of the reasoning proces, and, for instance, ODIN/TOR would also allow the example ES kinship to be a backward chaining system without redefinition of the rules.
3. Here I will rely on the SAM/SAMS theory of memory retrieval (Gillund & Shiffrin, 1984; Raaijmakers & Shiffrin, 1981) for some points. This theory is fairly mainstream, however.

REFERENCER

GILLUND, G. & SHIFFRIN, R.M. (1984): A retrieval model for both recognition and recall. *Psychological Review, 91,* 1-67.

LENAT, D.B. & FEIGENBAUM, E.A. (1987): On the thresholds of knowledge. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence, 2,* 1173-1882.

MERLEAU-PONTY, M. (1959/61): Le visible et l'invisible. Paris: Gallimard.

RAAIJMAKERS, J.G. & SHIFFRIN, R.M. (1982): Search of associative memory. *Psychological Review, 88,* 93-134.

RUBIN, E. (1920): Vorteile der Zweckbetrachtung für die Erkenntnis. In: Experimenta Psychologica. København: Munksgaard.

SCHÜTZ, A. (1932): Der sinnhafte Aufbau der sozialen Welt. Frankfurt/M.: Suhrkamp.

SLATTER, P.E. (1987): Building expert systems. Cognitive emulation. Chichester: Ellis Horwood.