Int. J. Heat Mass Transfer. Vol. 25, No. 9, pp. 1279–1289, 1982
Printed in Great Britain

0017–9310/82/091279–11 $03.00/0
Pergamon Press Ltd.

KNOWTRAN: AN ARTIFICIAL INTELLIGENCE SYSTEM FOR SOLVING HEAT TRANSFER PROBLEMS

AJAY SHARMA* and W. J. MINKOWYCZ
University of Illinois at Chicago Circle, Department of Energy Engineering, Chicago, IL 60680, U.S.A.

(*Received* 12 *October* 1981 *and in revised form* 30 *November* 1981)

Abstract—The detailed design specifications of an artificial intelligence system called KNOWTRAN are developed in this work. The design philosophy behind this system calls for a general and flexible program for acquiring, representing, storing and applying heat transfer knowledge. These ideas lead to the adoption of a knowledge-based approach to artificial intelligence programming. Furthermore, ideas about knowledge representation are developed to meet the requirements of a general heat transfer problem solver. This involves a hierarchical knowledge base managed by a flexible knowledge acquisition system. Finally, spacial representations are developed to accommodate objects and relationships in heat transfer problems.

NOMENCLATURE

BB,	blackboard;
DLI,	domain language interface;
ES,	explanation subsystem;
KAS,	knowledge acquisition subsystem;
KB,	knowledge base;
KBS,	knowledge-based system;
KNOWTRAN,	KNOWledge-based problem solver for heat TRANsfer;
KR,	knowledge representation;
PPS,	performance program subsystem;
SM,	system monitor;
SPB,	solution programs base.

INTRODUCTION

THIS PAPER contains a description of the goals, activities and the results of research on developing an artificial intelligence problem solver for heat transfer problems. The goal of this research is the design and implementation of a knowledge-based system (KBS) which will be expert in solving engineering problems involving heat transfer. The advantages of automating problem solving include lower cost as compared with human experts, reliable solutions, availability of a structured description of the solution technique used, access to a large and up-to-date knowledge base and ultimate use of the KBS as a research tool in exploring new techniques.

Before discussing the present project in detail, let us take a look at some of the previous work in application of artificial intelligence to problem solving. The pioneering work in this direction was done by Newell and Simon [1, 2] when they introduced their program "Logic Theorist" in 1956. Since then, many workers in artificial intelligence have worked on building computer systems to solve problems. The approach first taken by these researchers was to try to develop a general, domain-independent problem solver [2]. Although this work uncovered some basic techniques and limitations of artificial intelligence, it did not result in powerful, general-purpose problem solvers. One of the reasons for this was the size of knowledge base that is involved in solving practical problems. In other words, expert-level problem solutions are based on a large amount of contextual knowledge and experience that a human worker has accumulated.

The second approach adopted by researchers was to construct programs which limit their activity to a specific domain, and use a large amount of domain-specific knowledge to guide the problem solving process effectively [3, 4]. However, most of these programs have used *ad hoc* approaches for knowledge representation, which are too rigid to accomodate a variety of task domains. Nevertheless, these past efforts have resulted in some programs with impressive performance in, for example, internal medicine (MYCIN) [5], molecular genetics experiment design (MOLGEN) [6], symbolic mathematics (MACSYMA) [7] and mechanics problems in physics [8].

In the design of any knowledge-based system, the first and one of the most difficult problems to be tackled is that of knowledge representation [9–11]. In order to first understand and then solve a problem, one must be able to represent the objects, relationships and the abstract concepts of the domain in the artificially intelligent system. Without proper representation it would be difficult to achieve a significant amount of intelligence in the program, as the crux of "understanding" a problem is in abstracting it to a representation which is complete and amenable to manipulation and transformation that leads to a solution. Therefore, the initial thrust of the present research is directed towards

*Presently at IBM Corp., GTD East Fishkill, Hopewell Junction, NY 12533, U.S.A.

the development of appropriate representations. In addition, an effort is made to define the performance characteristics of all the subsystems in the heat transfer problem-solving system. While addressing the representation and system design problems, full consideration is given to the state of the art in the field of knowledge-based systems.

DESIGN PHILOSOPHY

The overall goal of this research is the design and implementation of an artificial intelligence system that acts as an expert in heat transfer. In order to further define this goal, consider the role that a human expert plays. He learns, solves problems and provides explanations on problem-solving techniques to others. In addition, some experts conduct research on their subjects to advance the frontiers of knowledge. The ultimate artificial intelligence system must do all that a human expert does. However, the state of the art does not yet allow one to create an artificial system with the breadth of knowledge and experience that humans have. Therefore, some restrictions have to be imposed on the desired performance.

Design goals

It was mentioned in the Introduction that development of a program with general problem-solving capability is not yet practical. The approach adopted here constrains the domain of expertise of the system to a specific area of engineering, which for the present is heat transfer. This restriction, however, is flexible in that the field of expertise only influences some of the subsystems in the design. This point will be discussed in detail later. As we are working on creating a heat transfer expert, let us call it KNOWTRAN (from KNOWledge-based problem solver for heat TRANsfer). Within its field – heat transfer – KNOWTRAN should be an equal to a human problem solver in that:

(1) given a problem for which its knowledge base is sufficient, KNOWTRAN should solve the problem with only as much input information as is required by a human expert;
(2) it should include facilities for interactive learning for initial, construction and subsequent expansion of its knowledge base;
(3) it should be able to explain the strategies and procedures that it adopts for solving the problem in order for people to have confidence in its solutions, and be able to provide a complete history of an unsuccessful solution attempt; and,
(4) interact with the user in the common language of the field that an expert can communicate in without any special training.

Tools and techniques

Having outlined the design goals, an in-depth analysis of what the goals entail is now presented. The first and last of the requirements set above are both fundamental and the most difficult to meet. In order for the system to be able to solve problems like people, it must recognize that problem solutions depend upon [12, 13]:

(a) the OBJECTS in the problem and their properties;
(b) the interactions between objects, i.e. the RELATIONSHIPS that exist among the various components;
(c) the CONTEXT of the problem, which determines the accuracy needed and the assumptions and approximations that can be made etc., and
(d) the known factors, and those that are desired, which together represent the crux of the initial problem statement.

An intelligent system is required to satisfy many other behavioral criteria in order to equal an expert [12, 13]. First, before actually attempting to solve a problem, the system must form plans of attack at various levels. The idea of planning in problem solving has been proposed by many researchers [14, 15]. Planning, which is also called "meta-level" inference by some authors, results in the narrowing down of possibilities to be considered for finding an actual solution procedure at an early stage. Moreover, in a system with a large knowledge base (KB), planning is mandatory for practical reasons like computer speed and memory limitations. It has also been shown [12] that planning is what separates experts from novices. In the context of KNOWTRAN, planning will involve things like classification of OBJECTS, RELATIONS and the problem environment. These classification activities result in pruning of further search options and, therefore, increase the problem solving efficiency. Further discussion of planning is postponed to a later section. Related to the idea of planning is the concept of hierarchical organization. A powerful technique of hierarchial object and relationship classification, representation and problem solution has been developed to support the KNOWTRAN system. These operational characteristics will also be described later.

Continuing with the discussion on expert-level problem solving, the next aspect to be considered is strategy. Psychology research seems to indicate [12] that experts deal with problems in a "forward reasoning" manner. That is, they start from the given facts and reason their way through known procedures to final solutions. Novices, on the other hand, start from the required solution and, using a backward chain of reasoning, tie it to the given quantities. KNOWTRAN design philosophy follows the path taken by experts, and forward search strategies are used as much as possible. This not only simulates human experts, but is the only strategy which, eventually, might lead to a system capable of original solutions to problems. An integral aspect of forward reasoning is using a knowledge base that is hierarchically organized and easily accessible at various stages of problem solving. This implies storage of knowledge about entire classes of problems, pointing to further detailed knowledge.

Complex problems are dealt with by first decompos-

ing them into simpler subproblems. It is at this decomposition stage that a lot of contextual knowledge comes to play. Some decompositions might require approximations and assumptions before they can be realized. The problem decomposition mechanisms might be required at any level of solution activity. One might need to decompose a problem right in the beginning and/or at later stages in finding a solution. This corresponds to an expert having a complete abstraction of both the current problem and of the domain knowledge in his mind at all times. In order to implement these ideas, a complex control strategy for KNOWTRAN is needed. Therefore, a good design for KNOWTRAN should include a *flexible* control system, which not only has a good control strategy built-in, but is also modifiable by a domain expert with no system reprogramming required. This design objective will be met by using meta-level knowledge for controlling KNOWTRAN operation.

To summarize, KNOWTRAN should start its problem-solving activity by first planning at various levels, operating on various entities (OBJECTS, RELATIONS etc.) using meta-level knowledge. Then it would proceed with detailed solution using a detailed knowledge base. All knowledge about control, planning and solutions is contained in the KB which should have a uniform knowledge representation structure.

Representation design

The most important decision in the design of an intelligent system is that about representation [11]. Because the structure of knowledge to a certain extent depends on the field, representations used are domain-dependent. Recently, some efforts have been made to find representations and structures which are generally applicable [11, 16], but these are still in preliminary stages of development. Moreover, we have set a design goal for representation which has more flexibility than any other proposed. KNOWTRAN representation scheme is such that all knowledge is uniformly, but flexibly structured. This goal will be achieved by including *representations of representation*, to any level required. In other words, KNOWTRAN will internally maintain structures that describe other structures that describe others etc., to any depth necessary. Thus KNOWTRAN will have knowledge about its own innards, to an extent not previously attempted in an intelligent system. This knowledge about itself results in one major benefit — the system is completely flexible as new structures can be defined by the user in terms of those already available in an interactive manner. This inclusion of new knowledge representations will be carried out in a natural dialogue, with intelligent help from KNOWTRAN derived from its existing knowledge base.

SUBSYSTEM SPECIFICATIONS

The purpose of this section is to describe a functional architecture for the KNOWTRAN system that satisfies the criteria set by the design philosophy. The description is conceptual in that certain functional subsystems may physically be integrated into one software module while other subsystems might actually be distributed over many program entities. However, the physical implementation need not be identical to the conceptual as long as, functionally, the system behaves in the manner described in this section.

A model for KNOWTRAN consists of eight major components which are as follows:

(1) the user subsystem,
(2) system monitor (SM),
(3) the knowledge acquisition subsystem (KAS),
(4) a knowledge base (KB),
(5) the performance program subsystem (PPS),
(6) a system blackboard (BB),
(7) explanation subsystem (ES), and
(8) the solution programs base (SPB).

These components make up the minimum KNOWTRAN system. Figure 1 schematically illustrates the organization of all these subsystems in KNOWTRAN. In order to add certain capabilities, like expertise in developing new solutions, the system will have to be further expanded. The questions about expansion will be answered in another section while here we proceed with a detailed specification of the present subsystems.

User subsystem

This component consists of the KNOWTRAN user and the domain language interface (DLI). KNOWTRAN will operate in two modes — the learning mode and the use mode. In the learning mode, the system interacts with an expert to build and expand its knowledge base, while in the use mode KNOWTRAN is used by people to solve problems. Problems are input to the program in either mode in the natural language of heat transfer. Similarly, other interactions are also in natural language (technical English) and the purpose of DLI is to translate between internal KNOWTRAN representations and English.

The design of DLI depends upon the common user language and the internal representations adopted. Translation between these two is guided by grammatical rules and by contextual knowledge [8]. There are many other factors to be considered but the detailed design of the DLI is postponed to the time when other subsystems of KNOWTRAN are more well defined. This delay is unavoidable because of the strong interaction between DLI requirements and the rest of the system. At this point we assume that a suitable DLI will translate user problem statements, queries and input knowledge, and KNOWTRAN solutions, questions and explanations. The final design of DLI will be guided by the work of other researchers [8, 17, 18] in natural language understanding and translation.

System monitor

The function of the KNOWTRAN system monitor is to control the interactions between other sub-

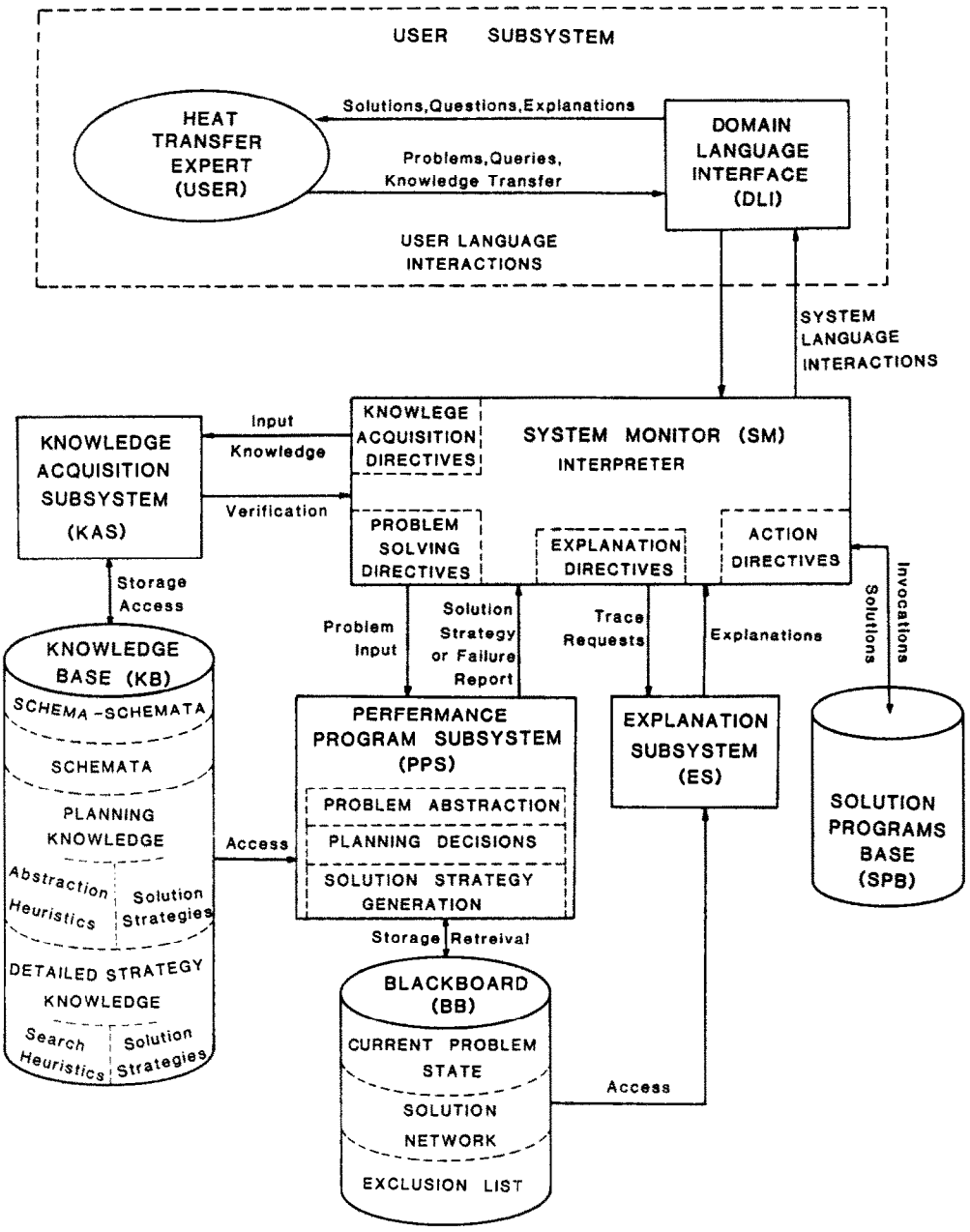

FIG. 1. Functional architecture of the KNOWTRAN system.

systems. Its role is primarily to maintain a uniform access and response protocol and to catch and process errors and interrupts. The system monitor is thus merely an interface between other routines. It has been decided, following the overall design philosophy, that all subsystems will interact with the system monitor in a uniform manner. By introducing SM as an integration tool, the rest of the KNOWTRAN design is made modular. Once more, the detailed design of the SM is deferred to a later stage in this project.

Knowledge acquisition subsystem

The system monitor will call upon the knowledge acquisition subsystem whenever KNOWTRAN is operating in the learning mode. KAS may also be invoked automatically when KNOWTRAN fails at any problem-solving task. It is through KAS that KNOWTRAN learns both new things and how to correct its mistakes.

KAS is designed for *interactive* transfer of expertise. It interacts with the heat transfer expert and transfers the expertise to KNOWTRAN knowledge base. This interactive transfer of knowledge will free the experts from the time consuming task of hand-assembling the KNOWTRAN knowledge base. An example of a KAS is the TEIRESIAS system [19] which was written to

automatically build and enhance the knowledge base of an intelligent system. The central theme of the TEIRESIAS program was the use of *meta-level knowledge* in assisting in construction and maintenance of a knowledge base. This was achieved by having the program "know what it knows". Thus, TEIRESIAS not only used its knowledge directly, but also was able to examine it, abstract it, reason about it and direct its application.

In writing the specifications for the KNOWTRAN knowledge acquisition subsystem, we start by including all capabilities of existing TEIRESIAS-like programs. Furthermore, we propose some important additional capabilities. The first extension is to require KAS to deal with multiple levels of knowledge representation rather than the two (meta-level and object-level) that TEIRESIAS had. Multiple levels are required because we want KNOWTRAN to be able to handle different kinds of knowledge structures. Previous artificial intelligence systems have been based on a single knowledge representation, for example production rules [5, 20, 21] or frames [11, 22]. KNOWTRAN design philosophy is to accommodate flexible knowledge representation in order to include as much of heat transfer knowledge as possible. This will be achieved by using schema (templates) for knowledge representations which start by defining a few representations in terms of system primitives. These initial schemas will be used in defining the next level representation (schema–schemata), and so on, to an arbitrary level of complexity. To understand this concept, consider the following analogy from conventional programming systems. In the FORTRAN language, high-level datatypes like REAL numbers are defined in terms of binary digits (bits) that are the primitives that a computer works with. The REAL numbers are then used to define COMPLEX datatype as a pair of REAL numbers. Therefore, theoretically, one has a hierarchy of datatypes defined in terms of primitive bits. Similarly, KNOWTRAN will work with representations that, at the bottom level, are all defined in terms of certain system primitives, and their definitions are known to the system itself in the form of the next higher level meta-definitions. Further details of the representation system adopted will be presented later.

Given the flexible representation philosophy, the specifications of the KAS become more demanding than those for previous programs. The KAS must insulate the user from the details of maintaining the complex hierarchical knowledge base while providing access to it in a flexible manner. These characteristics will be built-in in the KAS. Briefly, the KAS will be guided in its knowledge acquisition role by the history of previous solution attempts by KNOWTRAN. For example, if KNOWTRAN ever provides an unsatisfactory solution or fails to solve a problem, then the user will be given the choice of switching to the knowledge acquisition mode. Once that happens, KAS should provide a trace of the solution attempt and query the user about where erroneous decisions were made. Then, the expert will guide KAS in adding, deleting or modifying appropriate aspects of the KNOWTRAN knowledge base. These modifications will be made in an interactive manner similar to that of TEIRESIAS [19], except that the KNOWTRAN KAS will have many levels of meta-knowledge to work with. The details of the functioning of KAS will be explained in a following section.

Knowledge base

Most of the characteristics of the knowledge base have already been discussed in previous sections. Here we will bring all these ideas together and complete the specification of KNOWTRAN knowledge base. The knowledge base is central to the functioning of the entire KNOWTRAN system as *all the intelligence* or logic it requires to operate is contained in the KB. In other words, it will consist of all levels of meta-knowledge (schemata, schema–schemata etc.) which define the nature of knowledge contained, and the knowledge itself. Knowledge is also arranged in a hierarchy ranging from planning, abstraction and decomposition heuristics to the detailed problem solution strategies in heat transfer. Meta-level knowledge will be used by the KAS in initial construction and subsequent expansion of the knowledge base. In fact, meta-knowledge itself will be put in the KB by starting with a small nucleus of KAS-related meta-knowledge and then bootstrapping to a larger collection. Thus, one will build a system which is entirely KB-driven with very simple fixed code programs.

The knowledge base is thus merely a collection of structures which represent knowledge in the KNOWTRAN system. Therefore, the design specifications for knowledge representation also constitute the requirements to be satisfied by the KB, and these have already been outlined in preceding sections.

Performance program subsystem

Just as the KAS uses the meta-level knowledge to acquire heat transfer expertise, the performance program subsystem utilizes this knowledge to actually solve problems. In a way PPS is the heart of KNOWTRAN as it is this subsystem which will be invoked by the system monitor to work on a given problem. It will perform this function by applying the knowledge in the KB to the current situation while maintaining a record of actions taken, and checking the results of these actions to see if the solution(s) have been found. The most important design consideration for the PPS is to keep in mind that it must base all its decisions on what is contained in the knowledge base. This includes decisions regarding control, focus of attention, search strategy and actual solution approach.

In most of the previous artificial intelligence systems, some of the control information was embedded in the performance program [10]. This made these programs relatively inflexible in handling a variety of problems. Similarly, problem solution strategies like means–ends analysis [23, 24] or problem reduction

were inherent in the entire structure of some of the artificial intelligence programs. Other performance programs included planning ideas in their implementation [25–27]. The design objective set for the KNOWTRAN PPS is to strictly follow the doctrine of keeping all knowledge in the knowledge base, and so the PPS program itself will be relatively simple. It will merely fetch a chunk of knowledge from the KB and then take actions based on this knowledge. Thus, the implementation of planning and problem solution strategies will be in the contents of the knowledge base.

The blackboard

The KNOWTRAN blackboard is a conceptual entity which is used by the performance program to record all its activities while solving a problem. PPS will record its understanding of the problem statement, all subsequent knowledge application attempts, a trace of states that a problem has gone through, and a list of knowledge application failures on the blackboard. In practice, the BB will consist of various data structures, each tailored to the specific needs of storing one of these aspects of PPS activity. For example, the current problem state will be represented in a data-type chosen for problem description — a record of intermediate solution stages kept in the solution network (see Fig. 1) — and the exclusion list will be made up of a record of failures.

In addition to being a scratchpad for PPS operation, the BB will be accessible to the explanation subsystem for the purpose of providing feedback to the user as explained in the next section.

Explanation subsystem

This component of KNOWTRAN will be an important debugging and problem solving tool. The explanation subsystem will be invoked whenever the user asks KNOWTRAN for an explanation of a successful solution, or when a solution attempt fails. In the first case, the explanation request may be for verification of the solution, and in the second case, the information provided by the ES will be used in determining the reason for failure.

Many previous artificial intelligence systems have shown the value of a good explanation system [5, 28, 29] in increasing user confidence and as an important system debugging and expansion aid. The KNOWTRAN ES will work by accessing the BB to extract the relevant information and convey it to the user. This information can then be used by the user and/or the knowledge acquisition subsystem.

Solution programs base

Solution programs base will be a collection of programs which generate detailed problem solutions once the method of solution has been decided upon by KNOWTRAN. Typically, these will be numerical and symbolic mathematics packages that do the algebraic and numerical calculations for the problem. KNOWTRAN will use the SPB the same way as human problem solvers use a scientific subroutine library. The symbolic mathematics [8] programs will also be used at various intermediate stages of developing a solution strategy in reducing and decomposing the problem algebraically, and in other such operations. Numerical methods will be available to KNOWTRAN as a part of the SPB containing appropriate routines.

At the present state of the development of KNOWTRAN, the exact contents of the SPB are not important because all decisions regarding the solution method are based only on the knowledge base. KNOWTRAN need only be aware that a certain solution procedure is available and know the conditions of its applicability, together with what the procedure's output will be. These characteristics of the SPB will actually be part of the knowledge base and not of the SPB.

We have now sufficiently described the requirements imposed upon the design of each KNOWTRAN subsystem to be able to proceed with the actual design. This process is begun with the description of the KNOWTRAN knowledge representation in the next section.

REPRESENTATION

The primary goal of the initial research on KNOWTRAN is to develop the representations needed, in addition to setting the design goals detailed in the previous two sections. Before one can seriously think about the design of any intelligent system, one must decide about what representation is best suited to the domain of interest [30]. Although in a theoretical sense all representations are equivalent in that they are imbedded in the basic programming structures of the language being used, they are different in emphasis and utility. The differences in the usefulness of different representations are analogous to the differences between various programming languages, e.g., that between a high-level language like SNOBOL and machine language. Also, practically speaking, an artificial intelligence program will only be useful if its representation scheme is designed to be suitable for the kind of knowledge that characterizes the domain of expertise.

KNOWTRAN's representations are designed with the above factors in mind. Simply stated, the objective is to have a representation scheme which satisfies the following criteria:

(a) it should be as flexible and generally applicable as possible;

(b) there should be as much uniformity in knowledge representation for various types of knowledge as possible, and

(c) all aspects of the knowledge base should be accessible to the knowledge acquisition subsystem.

In the following sections we will describe a system for knowledge representation which meets these criteria better than any existing representation scheme.

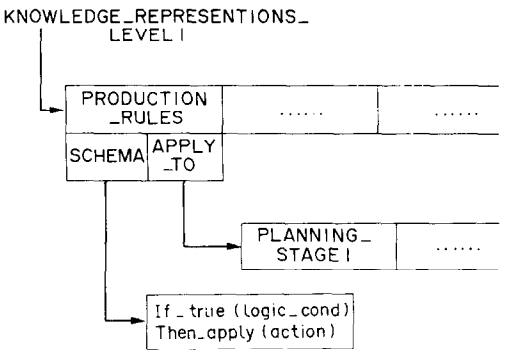

FIG. 2. Example of hierarchical representation.

Hierarchical representation

The basic ideas behind hierarchical representation were presented earlier. To recapitulate, hierarchical representation involves the inclusion of knowledge about how knowledge is represented in the KB itself. This goal will be achieved by starting with certain knowledge representation primitives and defining successive levels of knowledge representation (KR) in terms of lower levels. Figure 2 shows an example where "production rules" are defined in terms of KNOWTRAN knowledge representation primitives [If True (logic_cond), Then_apply (action)]. The data structure created is tagged to be a "schema" which implies that it is *not* a piece of applicable knowledge, but a description of a knowledge representation. In fact, this particular schema is input to the KAS which can then understand what a "production rule" is, and can then acquire knowledge which fits the production rule structure. A more complicated KR can now be defined in terms of the production rule schema and/or KR primitives. This is how the knowledge will be arranged in a vertical hierarchy.

In addition to the vertical hierarchy, the KNOWTRAN KB will be organized in an execution hierarchy which will control the application of knowledge. This and other details of KR are the topic of discussion in the following sections. Finally, it should be noted that the SCHEMA frame in Fig. 2 is analogous to a "program". KNOWTRAN uses the information in the second line of that figure only if the primitives in the first line succeed.

Objects

It is necessary to be able to express knowledge about objects in a problem and their properties. The object schema shown in Fig. 3 is an example of how objects might be represented in KNOWTRAN. Recall that with the flexible KR adopted, this schema shows only an example of object representation, and other schemata can be added in order to include objects which do not fit this one. Both KAS and PPS will not expect objects to be represented in any particular manner but will rather be guided by the object schemata.

In the example shown in Fig. 3, objects are represented by a data structure consisting of the object name pointing to the lists of its properties, relationships participated in, and a list of active classifications at various levels in the classification hierarchy. The properties list is initially constructed by the domain language interface as it analyses the problem statement. It is made up of property name (PNAME) plus value pairs with the values of unknown properties set to appropriate system primitives ("dependent variable", don't care" etc.). During the solution attempt, the properties list will be suitably modified by PPS.

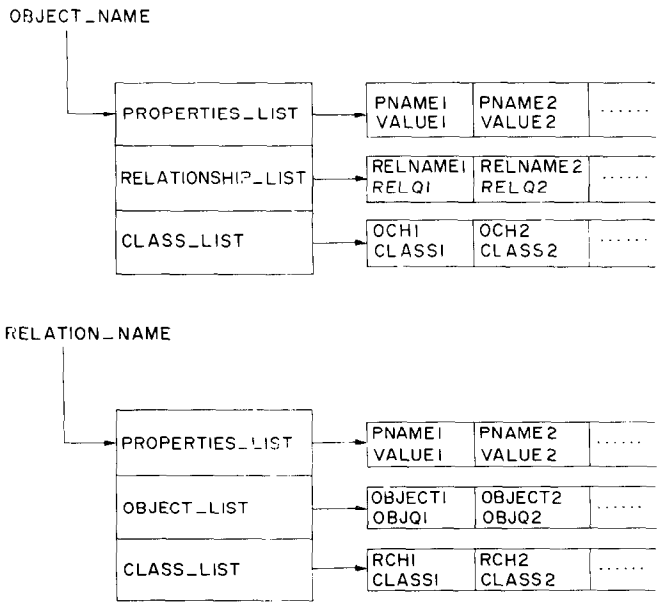

FIG. 3. Example object and relation representations.

The second component of object representation is a list of relationship names (RELNAME) of the relationships which involve this object. These relationships are the links between various objects and are described in the next section. Finally, the third component of the object representation is the classification list which is initially empty (NILL). As the PPS proceeds with the problem solution, it places the class names (CLASS) to which the object is found to belong at various levels of the object classification hierarchy (OCH).

Relations

Figure 3 also shows a representation suitable for expressing relationships which is similar to the object representation. Each relationship has a property list attached to it. As an example, consider a composite-slab conduction problem where the slab is made up of two materials. The objects in this problem will be the two different material slabs, with material properties like thermal conductivities attached to the OBJECT properties list, and boundary properties like temperatures attached to the RELATION properties list.

In addition to the properties list, the relation representation also has an object list and a classification list. The object list consists of the names (OBJECT) of the objects connected by this relation. In addition to the names of objects, the object list contains "qualifiers" (OBJQ) which qualify a relationship, e.g. in the 1-dim. slab problem the objects can be qualified as being "to the left of" and "to the right of" this relation. The relations list attached to the OBJECT representation also has similar qualifiers (RELQ). Finally, the classification list in a relation will be built up of classifications at various planning/solution stages, by the PPS [relation classification hierarchy (RCH)].

Control knowledge

KNOWTRAN performance program subsystem will be designed to be knowledge-base driven. The knowledge base will contain the information on PPS operation as "control knowledge". Initially, the control knowledge will be represented as meta-level production rules. These production rules will be defined as shown in Fig. 2.

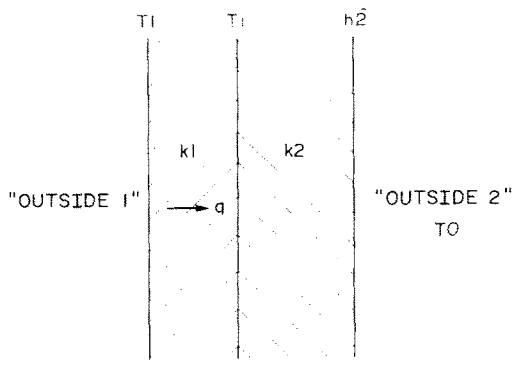

FIG. 4. Example heat transfer problem.

Planning and strategy knowledge

In addition to control knowledge, which determines the flow of control in KNOWTRAN PPS, the performance program will use many levels of planning and strategy knowledge. This knowledge is the actual domain-dependent knowledge which imparts "intuitive feel" of the subject to KNOWTRAN. The ideas for representing meta-knowledge have been drawn from previous work (e.g. [15]) and most of this knowledge will be represented as production rules and frames. The order in which this hierarchy of knowledge will be applied will be governed by PPS control knowledge.

Detailed problem solving knowledge

This portion of the KNOWTRAN knowledge base will be treated in the most flexible manner possible in that the decision regarding knowledge representation will be made on a case-by-case basis. KNOWTRAN KAS will guide the expert user in deciding upon a suitable representation for each chunk of detailed problem solving knowledge.

This knowledge acquisition process will begin with the KAS presenting the expert a "menu" of currently available knowledge representations (production rules, frames etc.) and asking if the knowledge can be represented in any of these ways. If that is the case, KAS will proceed to interactively assimilate the knowledge into known representations. Otherwise, the expert will have to create a suitable new representation using system primitives (see Fig. 2).

At this knowledge acquisition stage, the expert will also be asked to place the knowledge chunk (KC) at an appropriate level of the knowledge hierarchy. Once the KC is marked with its hierarchy level, the system will ensure that it is used at a suitable stage in problem solution.

KAS knowledge base

This portion of the KNOWTRAN KB will contain the information needed by the KAS for its own operation. It will use production rule representation (Fig. 2) to guide the KAS in acquiring knowledge, interacting with the user and in performing general housekeeping functions. It has been shown [19] that a KAS can be designed to operate based upon a production rule type meta-knowledge base. KAS knowledge will also guide it in ensuring that all new knowledge input to KNOWTRAN is properly assimilated in the system so that it is compatible with all programs and is tagged for use at the right stage of problem solution.

SYSTEM OPERATION

Having spelled out the basics of KNOWTRAN system design and knowledge representation, we can proceed with a description of how all these pieces will be put together operationally. In this section an

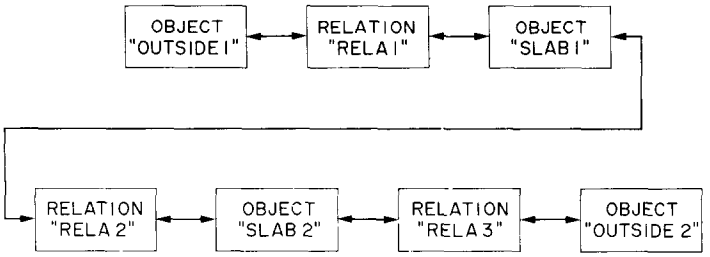

FIG. 5. Object–relationship network.

example heat transfer problem will be used to describe system operation.

Consider the problem of heat conduction in a semi-infinite composite slab (Fig. 4) with temperature $T1$ given for the left boundary and the heat transfer coefficient $h2$ and ambient temperature $T0$ known for the other boundary. Conductivities $k1$ and $k2$ are given, and the interface temperature Ti and the heat flux q are to be determined. This is a problem in 1-dim., steady state heat conduction.

The problem is first input to KNOWTRAN through the domain language interface of the user subsystem. Eventually, the DLI will have the capabilities of combined graphical and English input of problems of this type. From the problem statement, DLI constructs an object–relationship network consisting of objects and relationships in the problem. This network is shown in Fig. 5 and constitutes part of the KNOWTRAN blackboard (Fig. 1). The network consists of OBJECT and RELATION datatypes (representations) linked to each other by bidirectional pointers. In fact, the network shown in Fig. 5 is virtual in that it is constructed by storing appropriate values in OBJECT and RELATION datatypes rather than an independent network. While DLI is parsing the problem statement, it will create the OBJECTs OUTSIDE1, SLAB1, SLAB2 and OUTSIDE2, and RELATIONs RELA1, RELA2 and RELA3.

As an example of the kind of information put into these structures by DLI, Fig. 6 shows the detailed representations of RELA1, SLAB1 and RELA2 at this stage. The properties list of RELA1 has a single item — the known temperature $T1$ on it — and the classification list is initially empty. The notation P(OUTSIDE1) is used to denote that the object list of RELA1 is made up of pointers to object names and not the names themselves. Also, DLI attaches qualifiers like "LEFT_OBJECT" to the elements in the object list. These qualifiers will have a meaning to the PPS, expressed in its knowledge base. SLAB1 and

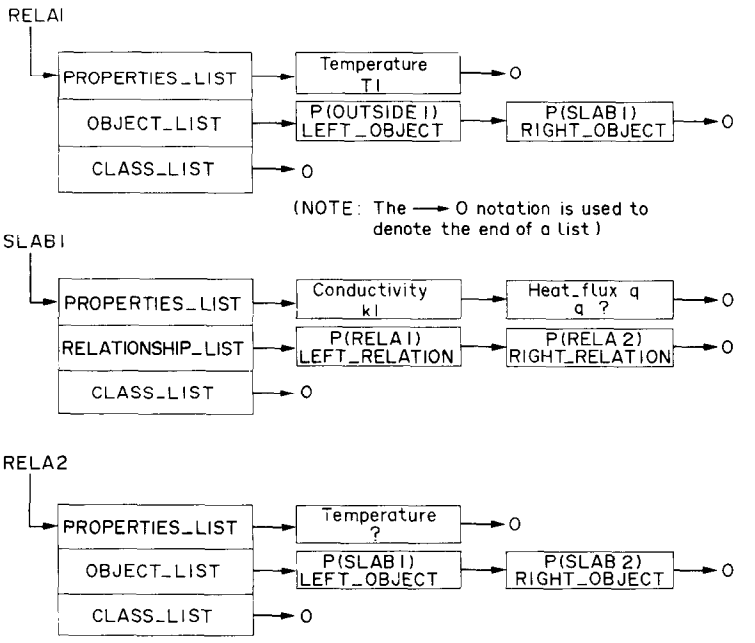

FIG. 6. Objects and relations in the example problem.

RELA2 representations are similarly constructed by the DLI to store information that it extracted while parsing the problem statement. Note that the property Temperature of RELA2 is given the value "?" which indicates an unknown that is to be determined during problem solution. Properties that are not known and not mentioned in the problem statements are not put on any property list. The other objects and relationships in the example problem (listed in Fig. 5) will also be represented by DLI in a manner similar to that shown in Fig. 6.

Once the DLI has parsed the problem statement, it passes control to the system monitor. The SM will ascertain the user's intentions — either to solve the problem or to instruct KNOWTRAN on solution techniques — and then take appropriate actions. If the user intends this problem to be an instructive example, then an appropriate SM flag will be set and, in either case, control passed to the PPS.

Upon initiating a problem solution attempt, the PPS will start a LOG_LIST of all actions taken, a FAILURE_LIST of unsuccessful trials and a SUCCESS_LIST of applicable knowledge chunks that caused an action leading toward a problem solution. PPS control knowledge will guide all its actions beginning with the application of planning knowledge. In the case of the example being considered, the planning knowledge rules will look at the object–relationship network and recognize the pattern for classifying the problem as a one-dimensional, steady state conduction. This classification will be attached to all object and relationship classification lists as the "planning level" classification.

Following the classification at planning level(s), the detailed problem solution knowledge will be used to solve the problem. In the case of this example, this may guide the PPS to construct an equivalent resistive electrical network and then invoke an electrical network solution program from the SPB to determine Ti and q. Thus a successful solution will be achieved and a detailed report on the solution can be obtained through an inquiry by the user. Such user inquiries will be routed to the explanation subsystem which will extract and translate the information contained in the blackboard as the object and relationship representations, the LOG_LIST, FAILURE_LIST and the SUCCESS_LIST.

If the problem was initially marked as an instructive problem, the difference in KNOWTRAN behavior will be that at each solution stage it will interact with the user At the beginning of an interaction, KNOWTRAN PPS will suspend operation and return control to SM which in turn will invoke the ES to provide an explanation of the latest actions taken. PPS will know that it has to proceed one step at a time because it will be passed a set "interactive" flag by the SM. This flag will also be passed to the ES so that it provides explanation of only the last step. Once the explanation is given, SM will invoke the KAS which will help the user interactively analyse the last action and suitably add to or modify the knowledge base to correct the mistakes, if any, made during the solution attempt. The manner in which KAS will operate will be a generalization of how TEIRESIAS [19] works.

If KNOWTRAN is given a problem that it cannot solve, it will go into a mode of operation similar to that that it adopts for instructive problems. PPS will suspend operation at the point that it cannot proceed any further from and returns control to SM with "can't proceed" flag set. This will again invoke ES and KAS to explain and overcome the difficulty encountered.

CONCLUDING REMARKS

In this paper we have laid out the detailed design specifications for KNOWTRAN — an artificial intelligence system for solving heat transfer problems. We have also developed the representation scheme for objects, relationships and heuristic knowledge for various purposes.

Further work on KNOWTRAN should begin with the coding of a nucleus mini-KNOWTRAN consisting of the SM, KAS, PPS and ES subsystems. Once this part is ready and debugged using either a dummy or skeletal SPB and direct input to SM, work can begin on the DLI. The domain language interface itself will be a major artificial intelligence project in natural language and graphical input processing. However, a good DLI will be essential from the usability point of view.

Acknowledgement—This work was funded by the National Science Foundation under grant ENG-7810318 and is gratefully acknowledged. The authors also acknowledge the assistance of Mr V. D. Doshi in various aspects of this project.

Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

1. A. Newell and H. A. Simon, The logic theory machine: a complex information processing system, *Inst. Radio Engrs Trans. Information Theory* **IT-2**(3), 61–79 (1956).
2. A. Newell, J. C. Shaw and H. A. Simon, Report on a general problem-solving program, in *Proc. Int. Conf. Information Processing*, Paris, France, pp. 256–264 (June 1959).
3. B. G. Buchanan, G. L. Sutherland and E. A. Feigenbaum, Heuristic DENDRAL: a program for generating exploratory hypotheses in organic chemistry, in *Machine Intelligence*, (edited by B. Meltzer and D. Michie) Vol. 4, pp. 121–157. American Elsevier, New York (1969).
4. N. J. Nilsson, Artificial intelligence, *Proc. Int. Fed. Information Processing Congress*, Stockholm, Sweden (August 1974).
5. E. H. Shortliffe, *MYCIN: Computer-based Consultations in Medical Therapeutics*. American Elsevier, New York (1976).
6. M. Stefik, Planning with constraints. Ph.D. thesis, Stanford University (1980).
7. R. Bogen *et al.*, *MACSYMA Reference Manual*. Lab. of Computer Sci., Mass. Inst. Technol., Mass. (1978).
8. G. S. Novak, Jr., Computer understanding of physics problems stated in natural language. Ph.D. thesis, University of Texas at Austin (1976).

9. M. J. Stefik, An examination of a frame-structured representation system, *Proc. Fifth Int. Joint Conf. Artificial Intelligence*, Cambridge, Mass (August 1977).
10. M. J. Stefik and N. Martin, A review of knowledge based problems solving as a basis for genetics experiment designing system, Report No. STAN-CS-77-596, Computer Sci. Dept., Stanford University, Calif. (1977).
11. R. Greiner and D. B. Lenat, A representation language language, *Proc. Am. Assoc. Artificial Intelligence First Nat. Conf.*, Stanford, Calif. (August 1980).
12. G. S. Novak, Jr. and A. A. Araya, Research on expert problem solving as a basis for genetics experiment *Intelligence First Nat. Conf.*, Stanford, Calif. (August 1980).
13. J. Larkin, J. McDermott, D. P. Simon and H. A. Simon, Expert and novice performance in solving physics problems, *Science, N.Y.* **208,** 1335–1341 (1980).
14. M. Minsky, Steps towards artificial intelligence, *Proc. Inst. Radio Engrs* **4**(1) (1961).
15. A. Bundy, L. Byrd, G. Luger, C. Mellish and M. Palmer, Solving mechanics problems using meta-level inference, *Proc. Sixth Int. Joint Conf. Artificial Intelligence*, Tokyo, Japan (1979).
16. R. Balzer, L. Erman, P. London and C. Williams, HEARSAY-III: a domain-independent framework for expert systems, *Proc. Am. Assoc. Artificial Intelligence First Nat. Conf.*, Stanford, Calif. (August 1980).
17. G. S. Novak, Jr., Representations of knowledge in a program for solving physics problems, *Proc. Am. Assoc. Artificial Intelligence First Nat. Conf.*, Stanford, Calif. (August 1980).
18. W. A. Woods, Transition network grammars for natural language analysis, *Comm. ACM* **13**(10), 591–606 (1970).
19. R. Davis, Interactive transfer of expertise: acquisition of new inference rules, *Artificial Intelligence* **12,** 121–157 (1979).
20. R. Davis and J. King, *An overview of production systems*, Report No. STAN-CS-75-524, Computer Sci. Dept., Stanford University, Calif. (1975).
21. J. S. Aikins, Representation of control knowledge in expert systems, *Proc. Am. Assoc. Artificial Intelligence First Nat. Conf.*, Stanford, Calif. (August 1980).
22. D. E. Smith and J. E. Clayton, A frame-based production system architecture, *Proc. Am. Assoc. Artificial Intelligence First Nat. Conf.*, Stanford, Calif. (August 1980).
23. G. W. Ernst and A. Newell, *GPS: A Case Study in Generality and Problem Solving*. Academic Press, New York (1969).
24. R. E. Fikes and N. J. Nilsson, STRIPS: a new approach to the application of theorem proving to problem solving, *Artificial Intelligence* **1,** 27–120 (1970).
25. P. H. Winston, Learning structural descriptions from examples, MAC TR-56, Mass. Inst. Technol. (1970).
26. G. J. Sussman and D. V. McDermott, Why CONNIVing is better than PLANNing, Mass. Inst. Technol., Technical Report MIT 255A, Cambridge, Mass. (1972).
27. E. D. Sacerdoti, Planning in a hierarchy of abstraction spaces, *Proc. Third Int. Joint Conf. Artificial Intelligence*, Stanford, Calif. (1973).
28. E. D. Sacerdoti, A structure for plans as behavior. Ph.D. thesis, Stanford University (1975).
29. B. G. Deutch, *Establishing context in task-oriented dialogs*, SRI Int. Tech. Note 114, Stanford, Calif. (1975).
30. P. H. Winston, *Artificial Intelligence*. Addison-Wesley, Reading, Mass (1977).

KNOWTRAN: UN SYSTEME D'INTELLIGENCE ARTIFICIELLE POUR RESOUDRE DES PROBLEMES DE TRANSFERT THERMIQUE

Résumé—On développe ici les spécifications d'un système d'intelligence artificielle appelé KNOWTRAN. La philosophie à la base de ce système demande un programme d'acquisition général et flexible, représentant, condensant et appliquant toute la connaissance des transferts thermiques. Ces idées conduisent à l'adoption de la programmation de l'intelligence artificielle basée sur la connaissance. De plus des idées sur la représentation des connaissances sont développées pour rassembler les besoins de résolution d'un problème général de transfert de chaleur. Ceci implique une base hiérarchisée de savoir pilotée par un système flexible d'acquisition de données. Finalement, les représentations spéciales sont développées pour approprier les objets et les formules dans les problèmes de transfert thermique.

KNOWTRAN: EIN PROGRAMMSYSTEM MIT KÜNSTLICHER INTELLIGENZ ZUR LÖSUNG VON WÄRMEÜBERTRAGUNGSPROBLEMEN

Zusammenfassung—In dieser Arbeit werden ausführliche Entwurfskriterien für ein Programmsystem mit künstlicher Intelligenz, genannt KNOWTRAN, entwickelt. Die Entwurfsphilosophie des Systems erfordert ein allgemeines flexibles Programm zur Aufnahme, Darstellung, Speicherung und Anwendung von Wärmeübertragungswissen. Diese Ideen führen zu einem auf der Kenntnis von Zusammenhängen basierenden Ansatz der Programmierung künstlicher Intelligenz. Weiterhin werden Vorstellungen zur Darstellung von Wissen entwickelt, um den Anforderungen eines allgemeinen Lösungsprogramms für Wärmeübertragungsprobleme (General Problem Solver) zu entsprechen. Dies führt zu einer hierarchischen Wissenbasis, die von einem flexiblen System zur Aufnahme von Wissen verwaltet wird. Schließlich werden besondere Darstellungsformen für die Erfassung von Objekten und Beziehungen in Wärmeübertragungsproblemen entwickelt.

НОУТРАН: ИСКУССТВЕННАЯ ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА ДЛЯ РЕШЕНИЯ ЗАДАЧ ПО ТЕПЛОПЕРЕНОСУ

Аннотация — Дано подробное описание искусственной интеллектуальной системы, названной Ноутран. Концепция такой системы требует универсальной гибкой программы сбора, представления, хранения и использования данных по теплопереносу. Использован метод базы данных для программирования искусственного интеллекта. Кроме того, разработан способ представления данных, который отвечает требованиям пользователя, решающего общие задачи теплопереноса. Он основан на иерархической базе данных, управляемой гибкой системой накопления информации. Наконец, разработаны способы представления данных, соответствующие объектам и соотношениям, встречающихся в задачах теплопереноса.