Tenth International Workshop

EXPERT SYSTEMS & THEIR APPLICATIONS

SECOND GENERATION EXPERT SYSTEMS

AVIGNON 90

May 28 - June 1st, 1990

AN INVESTIGATION OF THE ROLES OF PROBLEM-SOLVING METHODS IN DIAGNOSIS

William F. Punch III (1) & B. Chandrasekaran (2)

(1) Michigan State University
AI/Knowledge Based Systems Lab
Computer Science Department
A714 Wells Hall
East Lansing, MI 48824
USA

Tel. : +1 517 353 3541

(2) The Ohio State University
Laboratory for AI Research
Department of Computer Science
Columbus, OH 43210
USA

Abstract : This work centers on dynamic integration of multiple problem-solvers for the purpose of solving problems with broad and complex domains. A problem-solver based on dynamic integration uses diverse problem solving methods (neural net, rule-based, model-based, etc.) to reason about complex problems. Moreover, the sequence of method invocation is dynamic. By this is meant that method invocation depends on the present problem state and the capabilities of that method, not on a set sequence of invocation. This research is based on a preliminary model called TIPS (Task Integrated Problem Solving). TIPS provides a methodology in which the goal structure of a large-grained problem such as diagnosis is mapped to multiple problem-solving methods. A TIPS diagnosis problem-solver has been constructed in the domain of medical diagnosis (liver and blood disorders) which utilizes a number of different problem-solving methods. This paper will discuss the concept of a task-structure, the TIPS architecture and a medical diagnosis system implemented in TIPS. We will also discuss some of the shortcomings of the existing TIPS system and some approaches to solve these problems.

Keywords : integrated reasoning, task-structure, diagnosis

W.F. Punch would like to acknowledge the support of the Ameritech Foundation, B. Chandrasekaran would like to acknowledge support of AFOSR grants 87-0090 and 89-0250

1 Introduction

A common (though not universal) definition of the term *diagnosis* in a generic sense is: The mapping of signs and symptoms to malfunctions. Having made this definition, one must immediately note at least two inadequacies.

First, the above definition does not take into account how one may accomplish the diagnostic task. That is, there is no information on how the process of diagnosis should proceed. This is of course due to the fact that there are many views on how diagnosis should be done in the present field of AI. For example, heuristic/empirical/compiled approaches to diagnosis are based on a representation which pre-enumerates the malfunction categories and uses a reasoning method which searches through these categories for those that most clearly account for the observed signs and symptoms [6, 2, 15]. Other approaches which emphasize model-based approaches do not pre-enumerate the categories but determine malfunction based on representations using detailed models of the domain and reasoning methods like design models [9], malfunction/behavior modes[7] or simulation[8, 17].

The second inadequacy is the lack of inclusion of other aspects of diagnosis which, while not directly concerned with malfunction discovery, are often considered part and parcel of the diagnostic process. For example, the process of therapy is an important part of diagnosis as practiced in the real world. Another would be the process of data validation i.e., providing the diagnostic process with potentially validated data.

The problem seems to be that the generic process of diagnosis is a heterogeneous one. That is, the overall *task* of diagnosis can be decomposed into a structure of subtasks. This *task structure* [4] would be based on:

1. A goal structure for the diagnostic task.

2. A mapping that indicates which of those goals any particular method can achieve.

Thus one might define diagnosis not only in terms of the overall goal (association of signs and symptoms to malfunction), but also in terms of the richness of the subgoals one includes in diagnosis and the types of method used to achieve those goals. A diagnostic system realized in the above terms would have a number of advantages:

1. Such a system would have available multiple approaches for solving a similar problem. Thus the failure of one method does not mean failure for the whole problem-solver.

2. Such a system could potentially have more breadth in the types of diagnostic problems it could successfully solve.

3. The actual problem-solving process of diagnosis would depend on the availability of different types of diagnostic knowledge and on the available data.

Hence we can view diagnosis not only in terms of achieving a simple goal, but also in terms of breadth subgoals it can achieve and breadth of method used to achieve goals. There are a number of advantages to viewing problem-solving using a task structure:

1. It emphasizes problem analysis as an important part of system building. This means that it focuses on a view of the problem based on the various goals and subgoals of a problem and how they interact. The process of mapping goal achievement to various problem-solving methods is left to a later stage, thus concentrating on analysis of the problem and not on how to implement a problem-solver using any one method.

2. Once the goal structure is identified, problem-solving methods can be chosen to achieve the task goals based on their appropriateness for achieving that particular kind of goal. Thus multiple methods, each with a clear designation of what it can do, may be brought together in an integrated problem-solving environment.

3. Various goals may be achieved by a variety of methods. A system built using a task structure analysis allows experimenting with the effectiveness of various methods for achieving a particular goal in a larger problem-solving context.

4. Most importantly, such a task structure can be coupled with an overall control strategy to *dynamically* determine which goal(s) are appropriate to explore. Thus the sequence of method invocation is based on availability of knowledge and the problem-solving state, not on a fixed sequence. For example, if one domain has knowledge which indicates it should explore data validation, then that subgoal may be activated when the need arises. However, another domain may not have such knowledge available, in which case data validation will not be activated and alternate methods will be used to solve the overall problem.

1.1 An Example Task Structure for Diagnosis

One of the most important problems then is the creation of the goal structure for the overall problem. Consider the following medical case as a concrete example (taken from the 1954 case in [10]) of the kinds of goals that can be uncovered in a task such as medical diagnosis.

A 52 year old Italian laborer presents with the following symptoms:

- A series of "palpitations" that had associated chest pains and sweating. No overt signs of heart trouble.
- Blood test indications of anemia (low hematocrit, low rbc and low hemoglobin).
- Increased liver size along with an increase in bilirubin of the indirect (non-conjugated) type along with a slight jaundice, indicating a possible liver disease.

From the perspective of goals in diagnosis, the doctor as a diagnostician has a number of choices in reasoning at this stage:

1. Make the best diagnosis based on the available data, to wit, an unclassified type of liver disease and an anemia, probably *thalassemia minor* based both on the serological data and the patient's heritage[3].
2. Ask for more data that can help further the diagnosis. That is, given the present diagnostic conclusions, what data will give the most leverage in getting a more complete or satisfactory diagnosis? In this case, one could ask for more data about the liver disease, such as biopsy of the organ, or further blood studies to elucidate the kind of anemia.
3. Question data that does not fit in with the present best diagnosis and validate them if necessary. In the subsequent work on this case, the doctor ordered X-rays to confirm the existence of bleeding varices of the esophagus (bleeding from enlarged veins in the throat). Despite the fact that the X-rays were negative, other evidence was sufficient (vomiting of blood, stoppage of bleeding with a Blakemore tube) for confirmation.
4. Use the present diagnosis as a starting point for consideration of possible causal interactions among the affected organs that would explain the signs and symptoms of the case. In so doing, one may discover a relationship that indicates both a specific disease and data that confirms (or refutes) that disease. This creation of a "causal story" can be a powerful tool in paring down the possible diagnostic situations that need to be examined in more detail. In the case presented, a possible relationship exists between this specific type of anemia, which causes an excess of iron in the blood, and the liver which is responsible for maintaining iron equilibrium. This would suggest diseases of the liver based on overexposure to iron.

The following is an example a goal structure first developed in the domain of medical diagnosis[12] (in particular, Liver and Anemia related diseases). While no claim is made of this being a comprehensive goal structure for diagnosis, it is much broader in scope (in terms of sub-problems addressed) than most other diagnostic systems and is fairly generic is its applicability to other domains. The task structure for diagnosis is displayed in Figure 1.

Those goals are:

- **Do Diagnosis: The top level goal is to find a diagnostic explanation that covers all the findings and consists of hypotheses of the most detail possible.**

[3]Thalassemia minor is a common anemia of people with a Mediterranean heritage.

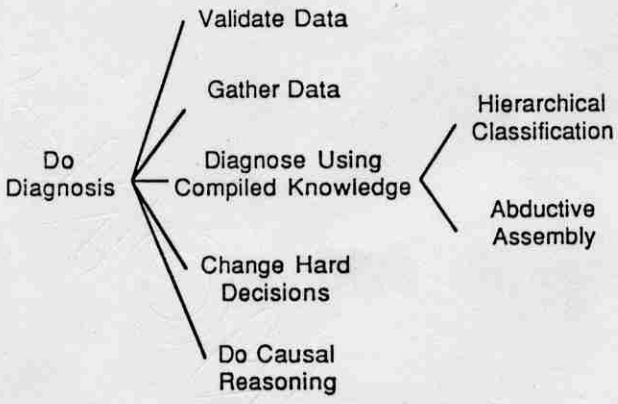

Figure 1: Task structure of the goals of diagnosis

- Diagnose using Compiled Knowledge: A subgoal of Do Diagnosis for creating a diagnostic explanation using compiled knowledge.
- Gather Data: A subgoal of Do Diagnosis for gathering data that facilitates making more detailed diagnostic explanations.
- Validate Data: A subgoal of Do Diagnosis for validating data that appears questionable.
- Change Hard Decisions: A subgoal of Do Diagnosis to explore alternate diagnostic explanations stemming from difficult hypothesis selections in Diagnose using Compiled Knowledge.
- Do Causal Reasoning: A subgoal of Do Diagnosis to determine the effects of proposed causal interactions between hypotheses in the diagnostic explanation.
- Hierarchical Classification: A subgoal of Diagnose using Compiled Knowledge to explore a preenumerated set of malfunction hypotheses that determines which are most plausible given the current problem state.
- Abductive Assembly: A subgoal of Diagnose using Compiled Knowledge to assemble a set of malfunction hypotheses which account for the current diagnostic findings and is causally coherent (i. e. , the hypotheses in the explanation do not conflict causally).

The *goal* of this research is to explore dynamic, task-integrated, problem-solving. The results of this work will be shown in two parts. The first part will discuss the TIPS (Task Integrated Problem Solving) architecture and its usefulness for task-specific integration. The second will briefly discuss a medical diagnosis system implemented in TIPS.

2 Architecture for Using Task Structure Analysis

2.1 Background

The motivation for this work comes from the experience of the authors with the *generic task* [3] approach to system building. Briefly, this view emphasizes the identification of domain-independent, generic approaches to solving particular kinds of high-level problems (classification, routine design). Once a task is identified, the problem is to discover forms of knowledge for both representation and control that are appropriate for that particular task. The promise of such an approach is that once a robust set of tasks have been identified, along with the tools that encode the knowledge for that task, they could form the building blocks of more complex problem solving tasks. That is, complex problem solving could emerge from the proper integration of generic task problem-solvers.

In the first generation of work on using generic tasks to build diagnostic and other complex problem solving systems, the invocation of a GT problem solver was pre-programmed as part of the task-structure. In other words, the invocation of GT problem-solvers had been implicitly "hard-wired" during the programming of the system. We realized that in order to use multiple methods to achieve problem solving goals we needed method selection to occur at run-time based on the state of the problem and availability of knowledge.

Thus the goal was to provide a general purpose architecture that allowed integration of high-level problem-solvers based on a dynamic problem state. The result was a way to analyze complex problems using the task structure (identification of goals in the complex problem and identification of capabilities to meet goals in each sub problem-solver) and an architecture that directly captured that analysis. This architecture is the TIPS architecture.

The TIPS approach to creating dynamically integrated problem-solvers is to provide only enough mechanism to allow monitoring of goal achievement and a mapping of methods that can achieve a goal. How these methods run is left to the designer as long as that conforms to a set of rules that indicate what it has done in terms of achieving its goal. This allows a knowledge engineer to take advantage of "tried and true" software (such as existing generic task problem solvers) without converting it to another format and allows a diversity of method and representation for different kinds of problem solving.

It is worth emphasizing again that the TIPS architecture is a response to the problem of dynamic integration from the point of view of generic tasks. As such, it has features that are particularly well-suited for integration of high-level problem solvers in response to a dynamic problem state. This is not to say that such an architecture could not have been realized in other existing systems. The discerning reader may note, after reading the sections on the actual architecture, some similarities between the TIPS architecture and other systems, in particular general blackboard systems such as BB1 [11] or the problem space architecture of SOAR[14]. This is neither troubling nor surprising.

Both BB1 and SOAR are general architectures and as such have a number of features which we do not need for our goal of integrating generic task problem solvers. Furthermore, as general purpose architectures they do not provide specific constructs/primitives for the knowledge engineer who wishes to construct a system for a high level task. The knowledge engineer must therefore construct/compose these task primitives and as a result is not provided with any task-specific constraints on using them. Recent work in both BB1 and SOAR has been concerned with providing higher level intefaces that hide some of the architecture level details from the knowledge engineer concerned only with programming and integrating high level tasks. TIPS is thus a proposal that shares important ideas with BB1 and SOAR about how to evaluate and choose methods for tasks at run-time. Because of its historical origin, namely the need to integrate generic tasks in a flexible way, its specific proposal for evaluating and selecting methods is somewhat different from those in BB1 and SOAR. The TIPS architecture used for diagnosis in particular makes a specific statement about the task-structure of diagnosis.

Thus we have no quarrel with other architectures. In some sense we have distilled through independent investigation the essential elements necessary to meet the particular problems already stated. In fact, we feel that we have much to give and learn from other designers since our approach to high-level integration dovetails well with research by groups such as SOAR and BB1 which investigate problem solving at an even more generic, broadly applicable though lower levels.

2.2 The TIPS Architecture for Problem Solving

The basis for the representation of control used in TIPS is the Sponsor-Selector system first used in DSPL [1] (Design Specialists and Plans Language). Such a structure is displayed in Figure 2. It consists of a hierarchy of three parts: a *selector*, some number of *sponsors*, and associated with each sponsor a *method invocation*. In short, the available methods are grouped under the selector, where each sponsor provides appropriateness measures for an associated method invocation. At any control choice point (i.e., some point in the flow of problem-solving at which another method could be invoked) the overall control process is to run all the sponsors to rate their associated methods, then have the selector choose the next method to be executed based on the sponsor values and other data.

2.3 Sponsors

Each sponsor contains information about when its associated method is appropriate. The sponsor evaluates this information and yields a discrete value indicating how appropriate the method is. The sponsor's knowledge is represented as a pattern match group similar to the knowledge groups of CSRL [2], the tool

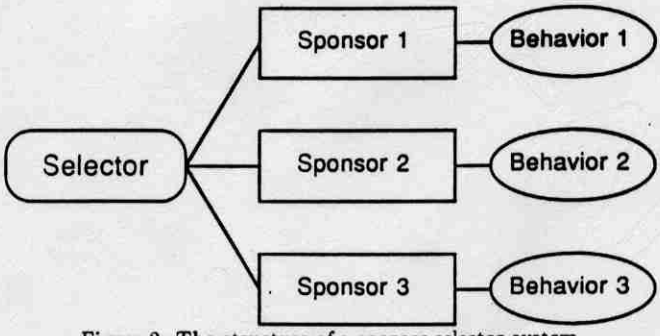

Figure 2: The structure of a sponsor-selector system.

used to build hierarchical classification systems. This representation is a table in which each column, except the last one, is a particular query about the problem-solving state of the system. Each row is a combination of answers to each question, a pattern of response. If a row "matches", that is, each pattern element of the row is true for the query with which it is associated, then the last element of the row is the symbolic value for that pattern match group. Thus the last column is a list of the symbolic values that could be potentially returned by the pattern match group. Control of row examination is of the simple "first true" type; that is the rows are evaluated in order until one of them matches. Thus the representation allows a knowledge engineer to investigate knowledge patterns in a specific order. Possible patterns of response are ruled out as the rows are evaluated until a matching pattern is found. If no patterns match and all have been examined, then some default value is returned, usually "unknown".

1) Has the classification method been applied yet ?

2) Is the present diagnostic explanation complete ?

Query 1	Query 2	Plausibility Value
F	?	3
T	F	1
F	F	-3

Figure 3: An example match group which determines the classification sponsor appropriateness value

Consider the example pattern match group of Figure 3 for the classification hierarchy sponsor. There are two queries: Query 1 represented by Column 1 is, "Has hierarchical classification been applied yet?" and Query 2 is, "Is the present diagnostic explanation complete?" (that is, explains all the data). Row 1 represents the pattern of response:

> If hierarchical classification has not yet reached completion, represented by the F or false answer to Query 1, then regardless of whether the explanation is complete, represented by the ? or "don't care" answer to Query 2, then return a 3.

The return values can be any discrete range of symbolic values to represent the appropriateness scale; in this case it is one of -3 (for totally inappropriate) to +3 (for totally appropriate).

Note that the pattern information represented in the rows can be more complicated expressions to match the kind of information returned by the queries, not just simple true/false distinctions.

There are two kinds of information available as appropriateness pattern match features:

1. Information about what methods have been applied so far in the problem solving process. These are questions about which methods have run so far, if at all, and when they ran. The architecture provides automatic access to these kinds of questions, such as: "Did the hierarchy run last?", "Has the abductor run in the last 3 cycles?", etc. Query 1 is an example of a query for this type of information.

2. Information concerning goal achievement, such as: "Is the explanation complete?" or, "Has this finding been explained?". This kind of information has to be supplied by the programmer who assembles both the behaviors and the pattern match knowledge and is not part of TIPS. Query 2 is an example of a query for this type of information.

2.4 Selectors

Each group of sponsors and their method invocations are grouped under a selector (see Figure 2). The selector does two things:

1. It organizes the set of methods that are available to be run on each cycle of evaluation.

2. It selects which method is next invoked based on the appropriateness ratings of the sponsors and other knowledge.

If a selector is activated, only the methods under it are available for evaluation on each selection cycle. Other groups of sponsor-selectors may be activated as a result of some method being evaluated, but only one sponsor-selector system at a time is responsible for selecting methods to evaluate.

A selector is responsible for choosing which of its methods to invoke. Its main criterion for so choosing is the appropriateness measures returned from its sponsors, but often this is not enough. The sponsor pattern match information *should* be coded with only local considerations in mind. In other words, it should not be the knowledge engineer's job to have to program each sponsor such that only one should be appropriate under all circumstances. Ideally, each sponsor is coded with little consideration of other sponsors with more global considerations of proper selection left to the selector.

The selector knowledge can be encoded in three ways, as shown in Figure 4.

1. The selection can occur simply based on appropriateness measures. If a clear behavior selection is available, i.e., there is exactly one highly rated behavior, then that method is invoked. If no clear selection is available and no other selection knowledge is available, then a random choice from among the best candidates is selected, i.e., those with the highest plausibility rating.

2. If there is a tie among the highest rated methods, tie breaking knowledge in the form of an *ordered priority list* can be used to break the tie. That is, the knowledge engineer can provide knowledge of method priority which determines which method is selected when appropriateness ratings are not

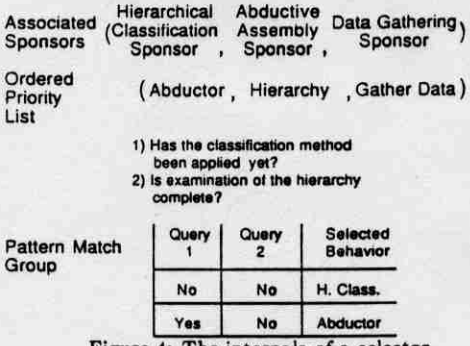

Figure 4: The internals of a selector

enough. If more than one sponsor is marked as appropriate, then the first appropriate method in the priority list will be chosen. As shown in Figure 4, if the hierarchy and the abductor are tied, then the abductor will be chosen.

3. For special situations, it is possible to override the normal choice mechanism with some pattern match knowledge similar to that found in the sponsors. If the pattern is matched, then the choice indicated in the matching row is used. If no match occurs, then the default priority list choice is used. The advantage of this is the processing time that can be saved to make a specific decision that is already determined by problem-solving analysis. The disadvantage is its inflexibility and narrow applicability. In the figure, if hierarchical classification has never been applied, as opposed to applied but not yet completed, then it should be the selected method. Notice this overrides the priority list.

The overall programming of the TIPS architecture is split into two parts:

1. Local decisions about appropriateness are coded in the sponsor in terms of knowledge about problem-solving progress and history of module activation, i. e. , both success/failure information and information concerning module selection.

2. Global decisions that concern actual choice of module invocation (based on sponsor results and other information) are coded in the selector.

As stated previously, multiple groups of sponsor-selectors can and often do exist, but only one such system is active at a time. Others may be activated by modules of a selector, but that one module-activated selector will remain the only active system until it completes.

Completion of cycling through a sponsor-selector systems is accomplished by a *Return* or *Fail* sponsor. These sponsors do not have any associated modules. Rather, they indicate when that particular sponsor-selector has finished its work or when it has failed to accomplish the task for which it was activated, that is whether the goal associated with the selector has been reached or not. These two sponsors often are at the highest priority since failure or completion are the two exit conditions. Setting of the Return and Failure sponsors are programmed just as any sponsor, by the knowledge engineer.

3 A Medical Diagnosis System in TIPS

In [12] a medical diagnosis system was developed in TIPS to demonstrate its usefulness. The system deals with a broad range of liver and blood diseases and their various interactions. The following briefly summarizes the modules used in the system, their roles and knowledge of when they are appropriate (see [12] for more details).

The diagnosis systems consists of the following modules:

- Compiled knowledge diagnostician. This module is responsible for creating diagnostic explanations using the abductive-assembly/hierarchical-classification architecture found in RED [16, 13]. It is used to meet the Diagnose using Compiled Knowledge goal.

  Briefly, this module has two subgoals, Hierarchical Classification and Abductive Assembly. Hierarchical Classification [2] is a problem solver that examines a hierarchy of (in this case) malfunction categories to determine the plausibility of malfunctions given the present circumstances. This plausibility information is used by the next phase, Abductive Assembly [13], to assemble a subset of the malfunction hypotheses such that the resulting *explanation* is most plausible, consistent and covers the findings.

  When Appropriate: This module is appropriate: as the very first module in a run (to create an initial diagnosis), if more data has been gathered, if a datum value was shown to be invalid and replaced by a validation method, or if causal reasoning has suggested an new malfunction hypothesis to consider for use in the composite explanation.

- Data gathering module. This module gathers evidence that can establish or rule-out pertinent diagnostic hypotheses. It is used to meet the Gather Data goal.

  In this particular case, the data that is needed next is determined by the nodes in the hierarchical classifier that suspended due to lack of data. If this data can be made available, then further exploration of the hierarchy is enabled.

When Appropriate: This module is appropriate when: the hierarchical classifier cannot explore leaf level hypotheses due to lack of data, or one of the potential hypotheses in a hard decision (see item "Redoing hard decisions" in this section) has an unanswered question in its specialist.

- Data validation module. This module checks the validity of certain data that have been questioned during the run of the compiled knowledge diagnostician module. It is used to meet the Validate Data goal.

  Described in some detail elsewhere (see [5]), the idea is the following. Rather than rely strictly on statistical "averaging" based on multiple sensor readings for one datum, this module uses expectations derived from partial hypotheses already formed by diagnosis. If these expectations are not met (i.e. a partial conclusion of "Liver Disease" requires some change in liver enzyme values and those values report in presently as normal) then there is a potential data problem. These unmet expectations are flagged and further investigated by specific test procedures that validate their value.

  When Appropriate: This module is appropriate if, in the process of diagnosis, a plausible composite explanation is generated whose data expectations are not met. Those data that do not meet expectation are noted as questionable and are submitted for validation.

- Redoing hard decisions module. This modules examines the assembly of alternate diagnostic explanations stemming from a hard decisions in the compiled knowledge diagnostician module. It is used to meet the Change Hard Decisions goal.

  A hard decision is a situation reached in abductive assembly when no clear criteria is available to differentiate among a set of hypotheses that offer to explain a finding. For example, the hypotheses A, B and C all offer to explain finding F and all have the same plausibility. If the only criteria available for differentiating which is the "best" hypothesis to explain F is plausibility, then there is no basis for a choice and a hard decision occurs. Typically the system makes a random choice and goes on but it also records the system state at that point so that the user may later go back and explore other possible solution paths.

  When Appropriate: This module is appropriate when a hard decision has resulted during the abductive assembly process.

- Causal reasoning module. This module applies the coherence view of causal modeling of [12] to the diagnostic explanations generated by the compiled knowledge diagnostician module. It is used to meet the Do Causal Reasoning goal.

  Briefly, associated with each diagnostic hypothesis is some generic knowledge about how the malfunction it represents modifies normal function. These changes are broadly categorized as functional changes (Anemia causes loss of oxygenation function), connective changes (blocked Common Bile duct causes back-up of bile) and structural changes (swelling of pancreas can push on the liver). If a possible link between the elements of a partial abductive assembly hypothesis are found, their relationship is explored using a model of the involved elements to determine some causal results not yet available to the compiled diagnosis component.

  Consider the example from Section 1.1. If hemolytic anemia causes a functional increase in iron products in the blood and the liver is responsible for clearing these products, a relationship between hemolytic anemia and the as-of-yet non-specific liver disease can be explored to discern a possible disease process. This can be done by simulating the liver model of "iron-clearing" with an additional parameter of excess iron. The results of this simulation may lead to a better understanding of the liver disease and suggest tests to validate the conclusion.

  When Appropriate: The causal reasoning module is appropriate when a possible causal interaction has been noted between hypotheses of the present explanation.

4 An example Case

The following is a brief example of the run of the diagnostic system on the case described in section 1.1. Many details have been left out but the trace is included to give the reader a feel for the type of problem-solving that goes on in the system. Note that depending on the initial data, the system may solve the problem in a completely different way.

The case is taken from a clinicopathological conference at the Albany Medical College (B-56469), May 6, 1954 and cited in Harvey and Bordley [10],

Step 1: Given the data base values and the findings to explain the hierarchical classifier/abductive assembler, which always runs first, comes up with an initial diagnosis of (**ThalassemiaMinor Hepatomegaly**).

Step 2: The explanation generated is rated via two criteria to determine if it is "good enough". The criteria in this case are that the explanation explain all the findings and that it consist of hypotheses that are most detailed, i.e., come from the leaf level of the hierarchy. This explanation is considered not good enough (the hypothesis elements explained all the findings but were not all at the leaf level of the hierarchy, i. e. , not an explanation of the greatest detail possible), so the sponsor-selector system is invoked. Most method selection will not occur through the action of the sponsor-selector mechanism.

Step 3: The sponsors of the TIPS architecture were evaluated (see Figure 3 for an example sponsor). The ones deemed appropriate this time were Gather More Data and Causal Reasoning, both of which were rated +3 on a scale of +3 (completely appropriate) to -3 (completely inappropriate). Gather More Data was appropriate since the hierarchy was not explored to leaf level. Causal Reasoning was appropriate because a functional relationship was noted between ThalassemiaMinor and Hepatomegaly. The relationship is based on the production of iron by ThalassemiaMinor and the normal liver operation of consuming iron associated with Hepatomegaly. The Causal Reasoning module was selected based on the tie-breaking knowledge of the sponsor's module priority list, shown in Figure 4, since causal reasoning has a higher priority than gathering data

Step 4: The causal reasoning module focuses on the functional relationship of iron, based on **excess-produce-iron** of **ThalassemiaMinor** and the **consume-iron** of **Hepatomegaly**. Note that the latter is a normal function of liver for which there is no evidence of change.

Step 5: Based on the proposed interaction of iron, a search is made for a model of iron-metabolism of liver. It is found in the Liver specialist associated with "iron-metabolism". The liver's iron-metabolism function is simulated on the initial conditions of excess iron, derived from the excess production of iron from **Thalassemia Minor** stored with its interaction knowledge. Note that it also searches for an effect of **consume-iron** on Thalassemia Minor, but finds none. This is important since the interaction could occur in either direction.

Step 6: The simulation yields the state of **iron-deposits-in-liver-cells**, which is noted as abnormal by the data base (the data base stores normal ranges for all data). The causal module searches to see if **iron-deposits-in-liver-cells** is associated with any known disease. It does so by searching another association list of states-to-malfunctions stored in the Liver specialist. An association is discovered between
iron-deposits-in-liver-cells and the disease **hemochromatosis**.

Step 7: The causal model runs the compiled knowledge of the hierarchical classifier associated with the disease **hemochromatosis** to determine if there is enough evidence to establish it. Note that this is an example of method invocation bypassing the sponsor-selector system. Such bypassing is appropriate here since the mechanism requires that such potential solutions be confirmed. However, it could also have been programmed through the normal sponsor-selection mechanism with no change in functionality. In any event, the node establishes so the disease **Hepatomegaly** is replaced as a hypothesis for the abductive assembler by **hemochromatosis** since the latter is a particular type of the former and the abductive assembler is trying to create the most detailed explanation possible. The causal reasoning module then returns control to the sponsor-selector.

Step 8: The explanation has not yet changed, the diagnostic explanation is still not "good enough" so the TIPS diagnostic architecture is again invoked. This time, and the Compiled Knowledge Diagnostician Module and the Gather Data Module are the only ones appropriate, both rated at +3. The Compiled Knowledge Diagnostician was appropriate because Causal Reasoning suggested hemochromatosis as a hypothesis for use in the composite explanation. The Gather Data Module is appropriate since the hierarchy was still not explored to leaf level. Since the Compiled Knowledge Diagnostician Module has a higher priority, it is invoked next.

Step 9: The Compiled Knowledge Diagnostician Module creates an alternate explanation with the elements (**ThalassemiaMinor hemochromatosis**). That explanation is complete and detailed, so diagnosis is finished and the case ends.

5 Evaluation

There are two aspects of the TIPS architecture that need to be improved. The first is the lack of any direct representation of the task structure goals. The second is the lack of any standardized guidelines of interaction between the methods and/or goals of a TIPS system.

The first problem is really one of representation. Figure 1 is a direct representation of the goals and their relationship in a problem such as diagnosis. In implementing this in a TIPS system however, those goals are only represented implicitly in the sponsor-selector system. In other words, there is no *direct* representation of the goals or of when the goals become active, or of when a goal has been achieved etc. At present, each sponsor must contain knowledge concerning when its goal(s) are appropriate for exploration and under what conditions its method is appropriate for achieving those goals. Moreover, this lack of clean separation leads to confusion on what gets sponsored. For example, Do Diagnosis is a goal but Hierarchical Classification is more of an approach. This is one of the more pressing problems that needs to be addressed as it does not conform to the task structure analysis.

The second problem is one of standardizing the means by which sponsors can monitor goal status and by which methods can indicate their success, partial success or failure. Likewise, a common means by which problem state information is gathered must be made available. At present, Lisp code specific to the goals and methods in the diagnostic system have been used but this is unacceptable for a general purpose architecture.

In short, we are addressing these problems by modifying the TIPS architecture to directly represent the task structure with a *goal tree*. The goal tree will provide direct representation of the task structure and standardize goal monitoring procedures for sponsors and methods.

References

[1] D. C. Brown and B. Chandrasekaran. Knowledge and control for a mechanical design expert system. *IEEE Computer*, 19:92–101, July 1986.

[2] T. C. Bylander and S. Mittal. CSRL: A language for classificatory problem solving and uncertainty handling. *AI Magazine*, 7, Summer 1986.

[3] B. Chandrasekaran. Towards a functional architecture for intelligence based on generic information processing tasks. In *Proceedings of the International Joint Conference on Artificial Intelligence 87*, pages 1183–1191. International Joint Conference on Artificial Intelligence, 1987.

[4] B. Chandrasekaran. Task structures, knowledge acquisition and learning. *Machine Learning*, 4:339–345, 1989.

[5] B. Chandrasekaran and W. F. Punch III. Data validation during diagnosis, a step beyond traditional sensor validation. In *AAAI87*, pages 778–782, 1987.

[6] W. J. Clancey. Heuristic classification. *Artificial Intelligence*, 27(3):289–350, 1985.

[7] J. deKleer and B. C. Williams. Diagnosis with behavior modes. In *AAAI89*, pages 1324–1330. Morgan Kaufmann, 1989.

[8] P. K. Fink and J. C. Lusth. Expert systsm and diagnostic expertise in the mechanical and electrical domains. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-17(3), May-June 1987.

[9] M. R. Genesereth. Diagnosis using hierarchical design models. In *Proceedings AAAI82*, pages 278–284. Morgan Kaufmann, 1982.

[10] A. M. Harvey and J. Bordley III. *Illustrative Case I in Liver Diseases*, pages 294–298. W. B. Saunders, 1972.

[11] B. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26:251–321, 1985.

[12] W. F. Punch III. *A Diagnosis System Using a Task Integrated Problem Solving Architecture (TIPS), Including Causal Reasoning.* PhD thesis, The Ohio State University, 1989.

[13] J. R. Josephson, B. Chandrasekaran, J. R. Smith, and M. C. Tanner. A mechanisim for forming composite explanatory hypotheses. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-17(3):445–454, 1987.

[14] P. S. Rosenbloom, J. E. Laird, and A. Newell. SOAR: an architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.

[15] E. H. Shortliffe. *Computer-based Medical Consultations:MYCIN*. Elsevier/North-Holland Inc., 1976.

[16] J. W. Smith, J. R. Svirbely, C. A. Evans, P. Strohm, J. R. Josephson, and M. C. Tanner. Red: A red-cell antibody identification expert module. *Journal of Medical Systems*, 9, Issue 3,:121–138, 1985.

[17] J. Sticklen. Integrating classification-based compiled level reasoning with function-based deep level reasoning. *Applied Artificial Intelligence*, 3(2):*in press*, 1989.