Managing Qualitative Simulation in Knowledge-Based Chemical Diagnosis

James K. McDowell and James F. Davis
Dept. of Chemical Engineering and Laboratory for AI Research, Ohio State University, Columbus, OH 43210

Deep knowledge about process behaviors plays an important role in the diagnosis of chemical processes. Cause-and-effect reasoning using deep knowledge is useful especially for interacting malfunctions. This work explores the integration of deep knowledge into task-specific, knowledge-based architectures for resolving interacting multiple malfunctions and presents a novel methodology called diagnostically focused simulation (DFS). Invoked in an auxiliary manner, DFS uses deep knowledge and performs qualitative simulation in a highly constrained manner. The close integration with other problem solvers is an evolutionary approach to using qualitative simulation in diagnosis and manages a normally computationally-explosive procedure. Diagnostic results from the compiled problem solver provide a situation-specific assessment of the chemical process, identify possible malfunction scenarios, and focus on appropriate levels of process detail. DFS effectively demonstrates a balance between run-time simulation and compiled problem solving in diagnosis.

Introduction

Knowledge-based techniques have emerged as a viable technology enhancing the development of aids to operators, supervisors, and/or process engineers for diagnosing malfunctions and abnormal process conditions in chemical process operations. This technology, as a basis for diagnostic methods in the chemical process domain, has been investigated considerably (Chester et al., 1984; Davis et al., 1985; Finch and Kramer, 1987, 1989; Shum et al., 1988, 1989; Venkatasubramanian and Rich, 1988; Ramesh et al., 1988; Oyeleye et al., 1989; Calandranis et al., 1990; Petti et al., 1990).

Included among these efforts in the chemical process domain and playing an important role in the construction of diagnostic knowledge-based systems, are problem-solving approaches based on a task-specific architecture (Shum et al., 1988; Ramesh et al., 1988; Shum et al., 1989). Rather than a feature of the chemical process domain, task-specific architectures are a theoretical viewpoint for constructing knowledge-based systems. Rather than a feature of the chemical process domain, (Chandrasekaran, 1983, 1986; Clancey, 1988; Steels, 1990). The basic premise of this viewpoint is that for a particular activity like diagnosis, knowledge for problem solving will take on specific and recurring forms. Recognizing these forms facilitates the gathering and structuring of knowledge for the construction and assembly of knowledge-based problem solvers.

This task-specific architecture expresses an effective domain-independent methodology for both robust and efficient run-time problem solving. Successful systems using task-specific architectures have been constructed in the chemical process, discrete manufacturing operations, and electronic domains for the diagnosis of single and independent multiple malfunctions (Shum et al., 1988, 1989; Ramesh et al., 1988, 1989; Myers et al., 1989, 1990; McDowell et al., 1990).

In the chemical process domain, however, it is well known that process and equipment malfunctions can interact via stream integration, control loops and sink-source relationships. These causally interacting malfunctions represent a relatively unexplored, but very important, application area for diagnostic knowledge-based systems. Causally interacting malfunctions include several possible scenarios. One might involve an equipment malfunction that causes an additional malfunction in a far removed part of the process. This scenario is realized in a diagnostic application for a fluidized catalytic cracking (FCC) unit. A malfunction in the temperature regulatory system of the FCC fractionator can potentially cause hot spots in the feed preheat furnace (Ramesh et al., 1989). Another scenario

Correspondence concerning this article should be addressed to J. F. Davis.

involves a malfunction interfering with the correct operation of a seemingly unrelated subsystem. This interaction scenario is detailed in a diagnostic application for a terephthalic acid process. One of the many examples of secondary interactions involves the pressure control system and the cooling system. Certain malfunctions in the cooling system will trigger an upset in the pressure control system, even though all the components of the pressure control system operate properly without malfunctions (Shum et al., 1988).

The resolution of such multiple interacting malfunctions requires the consideration of system structure and system behaviors in the context of the malfunction scenario. This type of knowledge commonly is referred to as deep knowledge, because the representation structures and inference procedures attempt to model the behavior of a physical system (Bylander, 1990).

This article presents a framework called diagnostically focused simulation (DFS) for using deep knowledge in knowledge-based diagnostic systems. The emphasis is on integrating deep knowledge with other forms of problem solving to fully resolve interacting multiple malfunctions, with the focus on the integration of deep reasoning into the task-specific knowledge-based system framework. A detailed description of the DFS framework forms the basis for subsequent reports of case studies and implementation.

As a background to these ideas, the first section reviews and classifies approaches that explicitly use deep knowledge in or as a basis for knowledge-based diagnosis. A common theme is that unconstrained reasoning with deep knowledge can be computationally expensive. This computational problem is remedied partially in these reported approaches through the integration of other kinds of problem solving. The first type, called *augmented* approaches, tends to augment the deep reasoner with a module that helps constrain the manipulation of deep knowledge during run-time. The second type, called *transformed* approaches, takes deep knowledge and transforms it into a different representation for use during diagnosis, thus avoiding the costly use of deep knowledge at run-time.

In the next section, our theory of generic tasks for diagnosis in the process domain is examined closely. The focus here is the process of compilation associated with generic tasks and the role of deep knowledge in the construction of diagnostic compiled problem solvers based on this paradigm. Task-specific architectures emphasize the knowledge representation and inference procedures that are common in problem-solving activities. Generic tasks form a template to organize knowledge, irrespective of the knowledge source. When deep knowledge is the source, the process of compilation transforms that knowledge into a problem-solving framework and defines a range of applicability for the compiled problem solver.

With this foundation in place, the DFS framework for integrating deep reasoning into the task theory is described. This conceptual framework is defined in terms of knowledge requirements and the inference procedures that are used to resolve specific cases of interacting multiple malfunctions.

As noted with other approaches that use deep knowledge, the deep reasoning aspect of the DFS task potentially is computationally explosive. To avoid this problem, the reasoning process in DFS is constrained at several levels:

- DFS is an auxiliary problem solver and as such is only invoked in specific cases. The primary activity of diagnosis (isolation and identification of malfunctions) is done by the compiled problem solver.
- The results of the primary compiled problem solver will suggest specific types of interactions among multiple malfunctions. These types of interactions suggest a specific simulation agenda with certain expectations concerning the simulation results.
- In addition to isolating possible interacting malfunctions, the compiled problem solver assesses the functional subsystems that are operating correctly. This assessment allows DFS to construct a multilevel view of the process, specific to the diagnostic case and suitable for cause-and-effect reasoning.

DFS attempts to reenact malfunction interactions by simulating a local malfunction, propagating its effect on the system behavior using qualitative simulation, and then evaluating any global effects on other malfunction hypotheses. The specific sequence of simulations is defined by the type of interaction suspected. DFS represents a novel approach to integrating deep reasoning and compiled reasoning. The integration is woven tightly and brings together important elements of both types of reasoning. The form of the integration is such that it cannot be classified as either augmented or transformed, rather it draws upon the features of each. The article closes with some concluding remarks concerning the integration of problem solvers.

Approaches for Integrating Deep Reasoning

Several available approaches to using deep knowledge and reasoning in diagnostic problem solving can be categorized as either augmented or transformed. Augmented approaches strongly separate the run-time use of deep knowledge from other types of problem-solving knowledge. Transformed approaches represent a form of knowledge compilation that uses deep knowledge and reasoning to generate malfunction states that are then captured into a new problem-solving architecture for diagnosis. Each approach will be discussed in terms of the modeling technique for deep knowledge, how the deep knowledge is used, and the form and content of its compiled component.

Representative of the augmented approaches is MODEX2 (Venkatasubramanian and Rich, 1988) and Grantham and Ungar's first-principles approach (FPA) (Grantham and Ungar, 1989). MIMIC (Dvorak and Kuipers, 1989) and MIDAS (Oyeleye et al., 1989; Finch and Kramer, 1989) represent the transformed approaches to integrating deep knowledge.

Augmented integration

Augmented approaches make a commitment to using deep knowledge directly at run-time during diagnosis. This problem solving involves causally searching through the deep model with the objective of generating valid paths from symptoms to root-cause malfunctions. To avoid this potentially computationally costly manipulation of deep knowledge, the approaches are augmented with other forms of problem solving that serve to direct or avoid the use of deep knowledge.

In MODEX2, the process structure is represented as components (process units) and connections (streams) using common object-oriented programming techniques. Deep knowledge about process behavior is represented using qualitative constraints and confluences (deKleer and Brown, 1984). During

diagnostic problem solving, deep knowledge is used in tracing the causal chain from symptom to symptom until the malfunction or malfunctions are isolated. This process can be very inefficient, especially if the malfunction is far removed from the initiating symptoms. The compiled component of MODEX2 attempts to avoid the use of deep knowledge. The compiled knowledge is in the form of associations, mapping symptoms directly to root causes. If the specific symptom to malfunction preenumeration exists in the knowledge base, then the malfunction can be isolated directly without using deep knowledge. Unfortunately, knowledge in the form of direct associations, though efficient, is rarely complete and very difficult to organize. Thus, MODEX2 often must resort to the use of deep knowledge that is applied in a strategy separate from the compiled problem-solving component.

FPA uses Forbus' process ontology (Forbus, 1984) to model the system behavior. Diagnosis using this deep representation is composed of a generation and a test phase. Generation involves tracing from symptom to cause in the qualitative physics model and proposing changes in the model that would account for the symptoms. A model change represents a candidate malfunction. The testing phase involves modifying the model and executing a form of inference called qualitative simulation to see if the model modification indeed accounts for the symptoms. Qualitative simulation is a predictive form of reasoning done on deep model representations to produce process behaviors. If the malfunction cannot account for the symptoms, then it is rejected. In FPA, both the generation and the testing phases are very inefficient and produce multiple candidates and multiple simulation results. To control this multiplicity, FPA augments the deep reasoning with a compiled problem-solving component. This compiled component is made up of strategies that attempt to control the computational burden of qualitative simulation. These strategies include rules of thumb for ordering, choosing, and testing malfunction candidates.

Transformed integration

This integration approach uses deep knowledge as the sole source of knowledge for their compiled diagnostic problem solvers. Malfunction scenarios are enacted from deep knowledge using qualitative simulations. The simulation results are organized into a problem-solving framework that is used for diagnosis. Transformed approaches avoid directly manipulating the deep knowledge at run-time and instead use a more efficient compiled representation for diagnostic problem solving. The compiled components of transformed approaches differ significantly from those of augmented approaches. The compiled components of transformed approaches are drawn directly from deep knowledge and replace, rather than augment, the direct manipulation of deep knowledge. The compiled components in augmented approaches do not have a formal basis in the deep representation and are derived independent of the deep knowledge. These features of transformed approaches are discussed in the representative examples below.

MIMIC uses Kuipers' constraint ontology (Kuipers, 1986) to model the process. Qualitative simulations are run for each malfunction scenario. The results are then processed using structured induction, an automated technique for feature classification. The product of structured induction is a decision tree, which forms the compiled problem solver in MIMIC. The tip nodes of the decision tree are the malfunctions, and the intermediate nodes are single symptoms. Connections in the decision tree represent possible values of the symptoms. Symptoms are requested one at a time, and depending on their values the decision tree is traversed in pursuit of the malfunctions. The decision tree is used during diagnosis, in place of directly manipulating the deep model.

MIDAS uses extended signed directed graphs (ESDG) (Kramer and Oyeleye, 1988) to model the behavior of process units. The process model is then transformed into a causal net structure, called an event network, which serves as the compiled representation of MIDAS for diagnostic problem solving. This network links malfunctions to a chain of intermediate symptoms and measurable symptoms. These preenumerated causal paths are used for diagnosis, instead of generating them at run-time. During diagnosis, this network is used to generate malfunction hypotheses that are later evaluated when sufficient symptomatic information becomes available.

Common features of the approaches

Although all the above approaches use deep knowledge explicitly, their choice of representation in each case is different. First, there is no unique representation for expressing knowledge about process structure and behavior. Secondly, all the approaches implicitly support the notion that unconstrained inference with deep knowledge is computationally explosive. All of these approaches integrate compiled problem solving in some manner (augmented or transformed) with the deep knowledge and reasoning for the specific purpose of controlling computational expense.

Task Approach to Diagnostic Knowledge-Based Systems

A distributed problem-solving framework has been formulated for the on-line malfunction diagnosis of complex manufacturing and process operations (Shum et al., 1989). This approach to knowledge-based systems, known as the generic task approach, was proposed originally by Chandrasekaran (1983, 1986, 1987). Each generic task makes a specific commitment to knowledge structures and inference processes. This forms a basis for capturing domain knowledge and organizing it into an effective problem-solving framework. The generic task theory has been applied successfully to building knowledge-based diagnostic systems in several engineering domains. Successful efforts in the chemical process domain include a terephthalic acid process and a fluidized catalytic cracking unit (Shum et al., 1988; Ramesh et al., 1988, 1989). A fielded system applying this approach to diagnosis of discrete PLC processes is described in Myers et al. (1989, 1990). This theory of knowledge-based systems has also been applied to the diagnosis of electronic devices (McDowell et al., 1990).

Compilation in task architectures

In these engineering domains, compiled diagnostic problem solvers have been constructed successfully using the paradigms of hierarchical classification (HC) and structured pattern matching (SPM). Compilation in the generic task theory refers

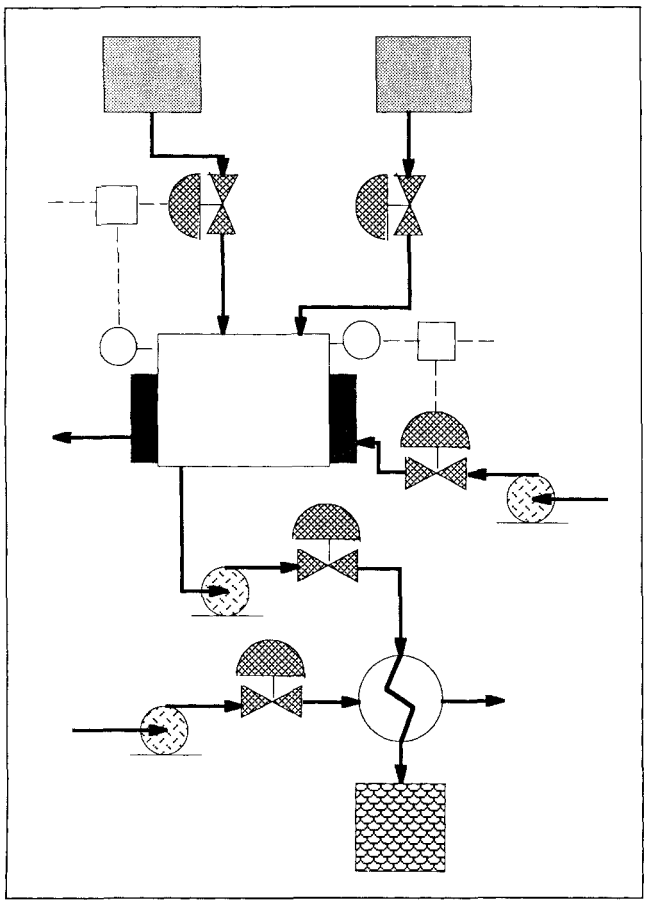

Figure 1. Chemical process.

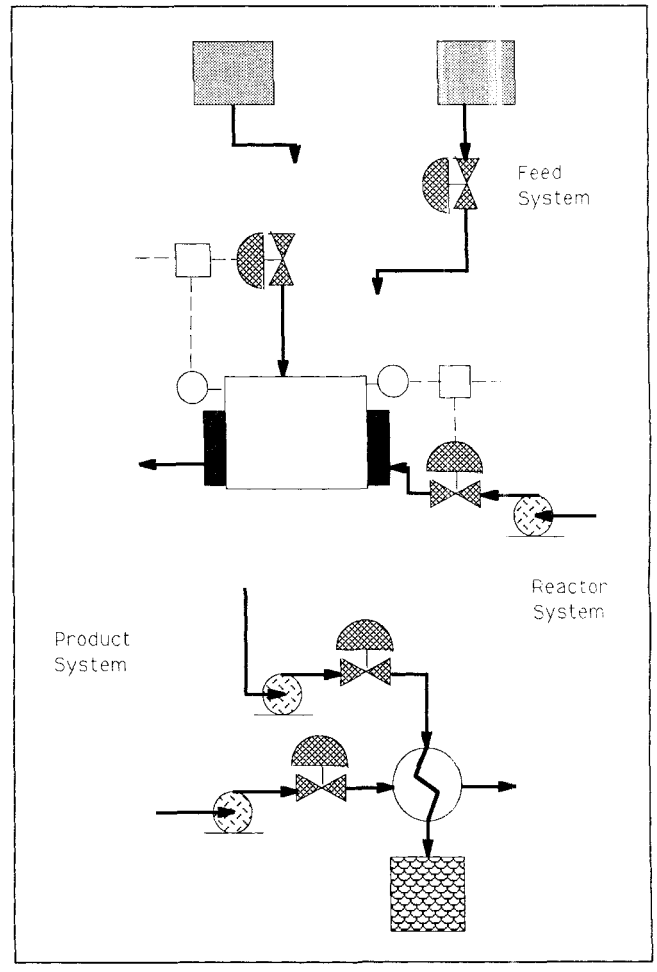

Figure 2. Initial decomposition of a chemical process.

to a commitment to problem-solving architectures without any reference to the source of knowledge. The type of problem solver serves as a template for organizing knowledge. The term "compiled" often has been used interchangeably with terms like "heuristic" (Venkatasubramanian and Rich, 1988; Fink and Lusth, 1987). Our view, however, is that the term heuristic implies a commitment to a source of knowledge and is very different from compilation. A problem-solving framework might be encoded with knowledge from several sources: deep behavioral understanding, expert experience, and process history. Given this viewpoint, the issues of knowledge source and run-time problem solving seem to be orthogonal (McDowell et al., 1989). In the course of examining these task architectures with respect to deep knowledge, the features of these compilation commitments are examined closely.

A key element in the construction of the hierarchical knowledge structure used in HC involves the identification of functional systems and subsystems in the process or device being diagnosed. This process or device could be a chemical process, a manufacturing process, an electronic device, etc. Figure 1 was adapted from a process example given by Venkatasubramanian and Rich (1988). This chemical process is composed of several common process units such as pumps, valves, piping, a reactor, and heat exchanger. The procedure of identifying functional systems/subsystems decomposes the chemical process into a malfunction hierarchy. Figure 2 shows the decomposition of the chemical process into its major functional systems: feed system, reactor system, and product system. The decomposition procedure is continued in Figures 3 and 4. The final result of this procedure is the functional hierarchy shown in Figure 5. The nodes in this hierarchy represent malfunction hypotheses. The tip-level nodes represent the process units

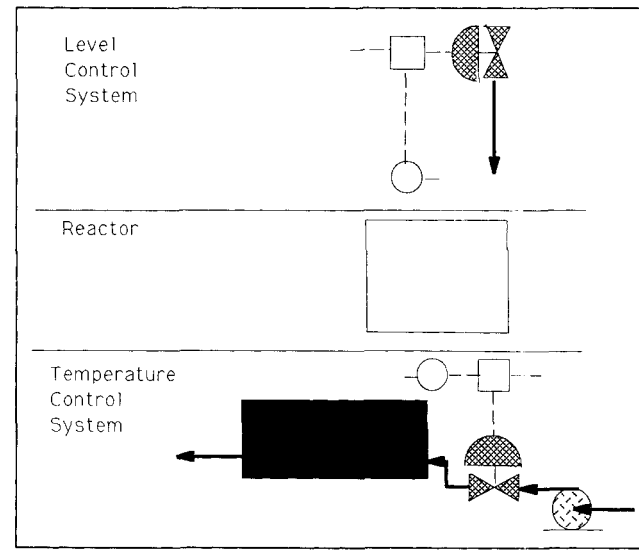

Figure 3. Decomposition of the reactor system.

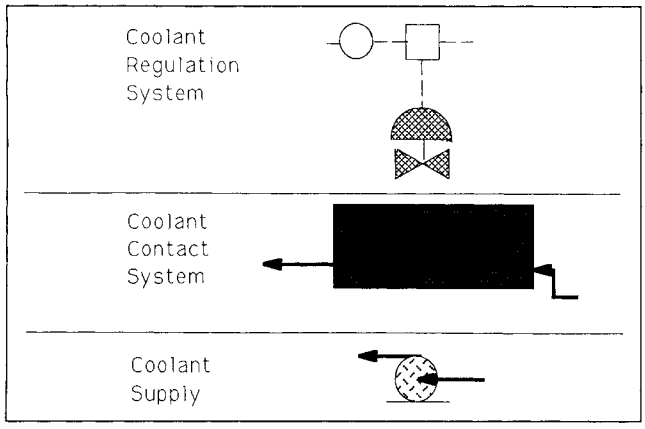

Figure 4. Decomposition of the temperature control system.

themselves and the intermediate nodes represent malfunction abstractions. As demonstrated, the procedure of constructing the hierarchy is one of trading process connections for connections in a problem-solving framework. At the completion of the procedure, the process structure is no longer explicit. The purpose of this compilation is to organize knowledge into a useful form for a particular task. The compilation procedure for HC organizes the process behavior in terms of connections in a malfunction hypothesis hierarchy. Diagnostic search operates on this problem-solving structure and not on the chemical process connectivities.

As shown in the generation of Figure 5, HC attempts to organize the process malfunctions into a problem-solving framework. The nodes of the hierarchy then represent malfunction hypotheses at successive levels of detail. Such a problem-solving framework offers an advantage during diagnosis. If a malfunction hypothesis high in the hierarchy can be rejected, then consideration of malfunction hypotheses below this node is not necessary. This inference procedure prunes the search space of malfunctions and only pursues the relevant branches of the hierarchy, until a tip-level node or nodes are established. A great deal of focusing can be generated in this way.

Each node in the malfunction hierarchy represents a malfunction hypothesis. The evaluation of such a hypothesis requires different knowledge and inference procedures and represents a generic task different from that of searching the hierarchy itself. This task used in evaluating malfunction hypotheses is called structured pattern matching. In this task, specific symptomatic features are accessed and matched against predefined patterns. Each pattern has associated with it a score or rating called a confidence value. Common in all diagnostic hypotheses are the values "confirmed" and "ruled out." When symptomatic information matches a pattern of features, the appropriate confidence value is assigned to the malfunction hypothesis. An example of the knowledge organization for this task is shown in Figure 6. Each node in the malfunction hierarchy can have one or several such pattern matchers associated with it, which can be organized hierarchically. Compilation of this form eliminates the need for generating symptom patterns at run-time.

Another form of compilation relevant to HC and SPM involves the elimination of unnecessary causal links. If a malfunction causes a chain of symptoms to arise, for example, A(malfunction) → B(symptom) → C(symptom) → D(symptom) and if knowledge ensures that this causal path always occurs, then tracing the intermediate causal links is not necessary. The elimination of such causal links is useful in knowledge organization and selection for both HC and SPM.

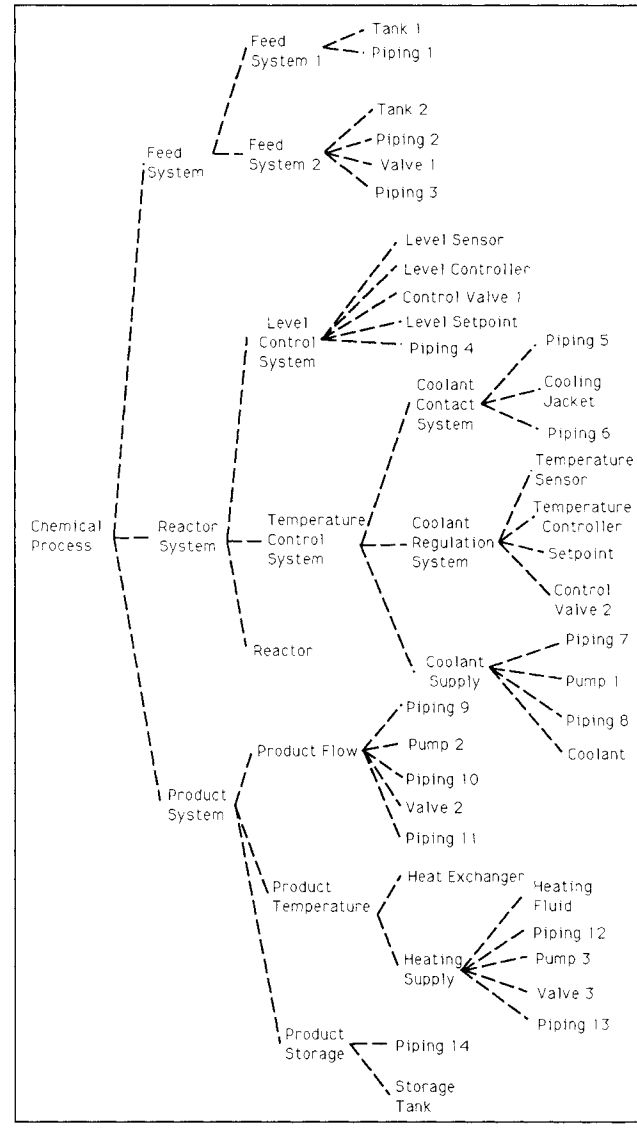

Figure 5. Malfunction hierarchy for a chemical process.

For diagnostic systems that make use of HC and SPM, we have thus identified three forms of compilation:

1. Identification of system/subsystems
2. Selection of key symptomatic information for evaluating diagnostic hypotheses
3. Elimination of intermediate causal links.

Our past efforts in building knowledge-based diagnostic systems have demonstrated that for processes or devices where a complete understanding of the diagnostic situation is available, constructed diagnostic systems are completely robust. By robustness we mean that the diagnostic system always was able to arrive at the correct diagnostic conclusion. These situations involved single and independent multiple malfunctions in processes where the certainty in selecting symptoms for establish-

POTHESIS: Malfunction in Pressure Control

FEATURES:

F1: Pressure alarm above condenser activated?
values: (yes, no, uncertain)

F2: Pressure measurement above condenser?
values: (normal, high, low)

F3: How far open is the pressure control valve above the condenser as indicated by the valve positioner?
values:(normal, high, low)

HYPOTHESIS CONFIDENCE RATING:

F1	F2	F3	CONFIDENCE RATING
yes	low	high	confirmed
yes	high	low	confirmed
?	low	high	very-likely
?	high	low	very-likely
?	low	low	rule-out
?	high	high	rule-out
NO MATCH PATTERN			very-unlikely

Figure 6. Pattern matching table for hypothesis evaluation.

ing malfunctions were very high. In the cases of multiple interdependent malfunctions, HC is able to robustly identify potential interacting hypotheses, but is not able to bring the situation to complete resolution by identifying the root causes.

Differences in compilation approaches

As previously discussed, the transformed approaches to integration do not use deep knowledge during the diagnosis process, but use compiled problem solvers that differ from the task architectures. MIMIC's decision tree makes a commitment to feature classification that may be used for diagnostic problem solving. There is, however, no explicit system/subsystem decomposition or malfunction hypothesis evaluation. The intermediate nodes in the decision tree relate only to features (symptoms) and are not intermediate malfunction abstractions. Only the tip nodes of the decision tree represent the malfunctioning system units as seen in the functional malfunction hierarchy used in HC. MIDAS's causal net structure does not clearly evaluate any system/subsystem decomposition or malfunction hypothesis. Nodes in the causal net are either root-cause malfunctions or symptoms that arise due to the malfunction. An additional difference in task compilation is that sources other than deep knowledge can be used in the problem-solving architecture. As discussed earlier, the transformed approaches make a specific commitment to their deep knowledge as the sole source for the diagnostic problem solvers.

The compiled problem-solving component of the augmented approaches does not have the organization of the transformed or task approaches to compilation, because it relies on the manipulation of deep knowledge for most of the diagnostic problem solving. The compiled problem-solving component serves as a diagnostic shortcut or provides rules of thumb for constraining the deep reasoning. Recall that MODEX2 uses compiled knowledge in the form of associations that map directly from symptoms to malfunction. FPA's compiled component is committed to constraining the manipulation of deep knowledge by controlling the generation of diagnostic candidates and providing ordering for candidate testing.

This is not to say any of these problem-solving systems would lack in robustness, since robustness depends on the source of knowledge. It does point out, however, that there are different ways to use deep knowledge for diagnosis, both in a compiled form and directly at run-time.

Compilation with deep reasoning

Task-specific problem solvers using HC/SPM in the process domain tend to represent systems with a high level of decomposition. The abstract malfunction hypotheses are evaluated in a local manner, drawing on knowledge related only to confirming or ruling out that specific hypothesis. These compiled systems are effective in identifying single malfunctions and independent multiple malfunctions. This class of malfunctions can be organized in terms of a functional decomposition and local consideration of symptomatic information. Based on our observations to date, single and independent multiple malfunctions cover the majority of scenarios in the chemical process plant domain. The commitments of compilation do not affect the robustness of the diagnostic system for these malfunction scenarios.

The class of all malfunctions can have causally related multiple malfunctions. Interacting situations are possible via stream integration, control loops, and sink-source relations. Thus, multiple-interacting malfunctions are a possibility and a concern in the chemical engineering domain.

The resolution of multiple-interacting malfunctions requires the consideration of the system topology and the relationships among process variables in the context of a malfunction. This would suggest that the direct manipulation of deep knowledge would be necessary in such a case. Though all the systems previously discussed use deep knowledge, only MIDAS considers the area of causally interacting malfunctions and then in the narrow area of induced sensor failure. The remaining systems, MIMIC, MODEX2, and FPA, are committed to single malfunctions and independent multiple malfunctions. They may be able to identify cases of interacting multiple malfunctions; however, it would be difficult to distinguish from the case of independent multiple malfunctions, unless the interaction was preenumerated.

The focus of this work is to significantly build upon the task-specific approach in the area of causally interacting multiple malfunctions. The intent is to leverage information contained in the compiled structures of HC, but is not found in the augmented or transformed structures of other approaches. The functional decomposition contains information to con-

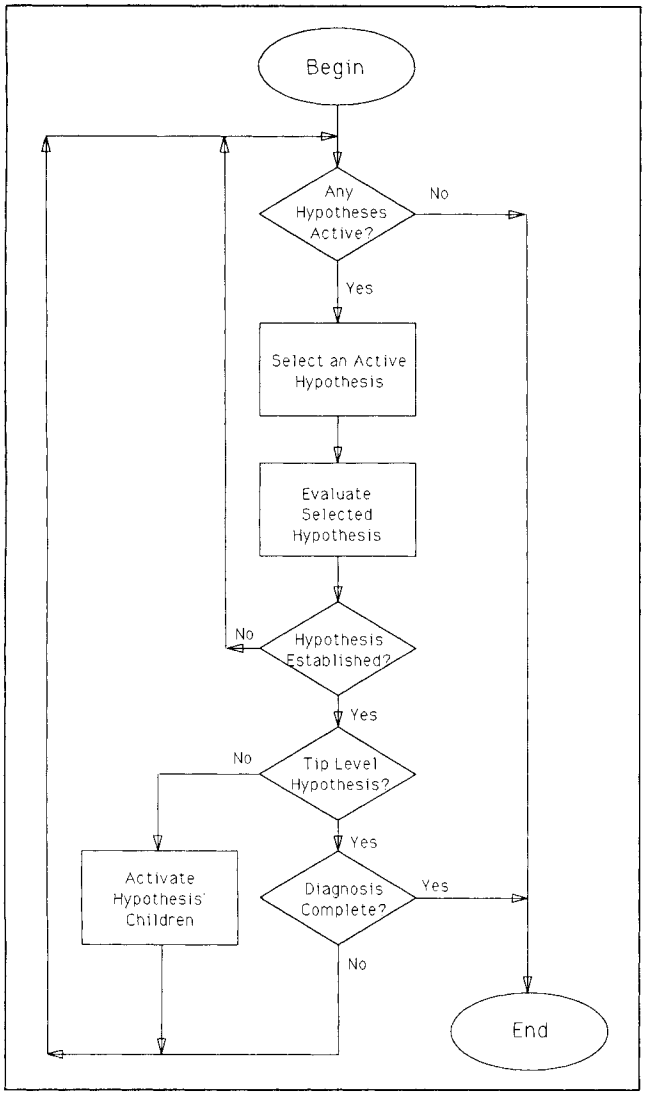

Figure 7. Establish-refine inference procedure.

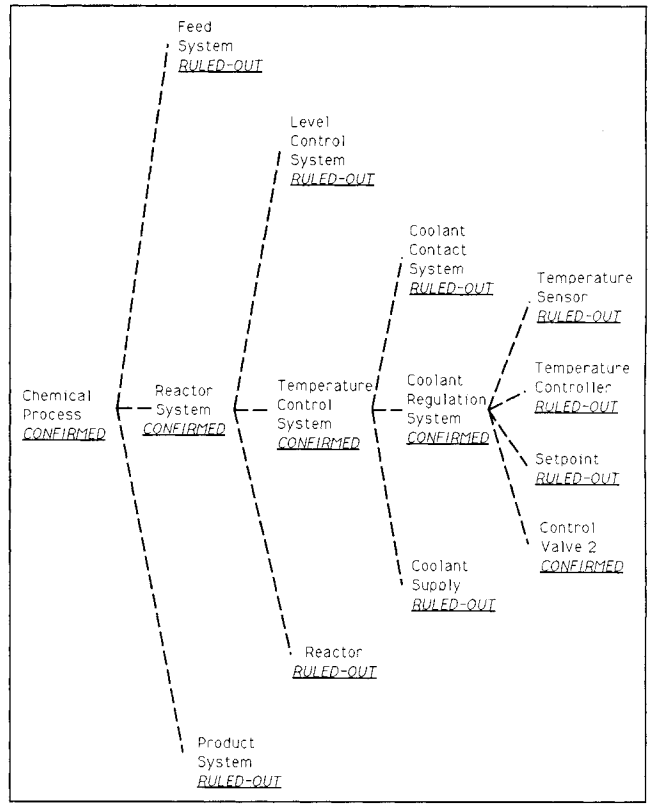

Figure 8. Sample confidence value hierarchy.

struct the case-specific system structures, along which multiple malfunctions may interact. From a generic task view, this requires an additional auxiliary problem solver for manipulating deep knowledge. As part of the task architecture, diagnostically focused simulation can be viewed as an auxiliary information processing task that involves reasoning about cause-and-effect among process variables across a process topology for the specific purpose of resolving uncertain diagnostic conclusions relating to multiple-interacting malfunctions.

Details of Information Processing in DFS

Because DFS is an auxiliary problem solver, its processing is invoked after the compiled problem solver completes its diagnostic assessment. Figure 7 illustrates the inference procedure that is used by the compiled diagnostic problem solver. The top node in the hierarchy is made active and the procedure in Figure 7 begins. Specific symptomatic information is examined in the context of evaluating a selected malfunction hypothesis. If the hypothesis is established (this usually requires a confidence value score of "confirmed" or "very-likely"), then its children in the hierarchy are made available for possible evaluation. When a malfunction is rejected, the portion of the hierarchy below the hypothesis is pruned effectively by the inference procedure. One or several tip-level node malfunctions are established in the normal procedure of diagnosis. The re-

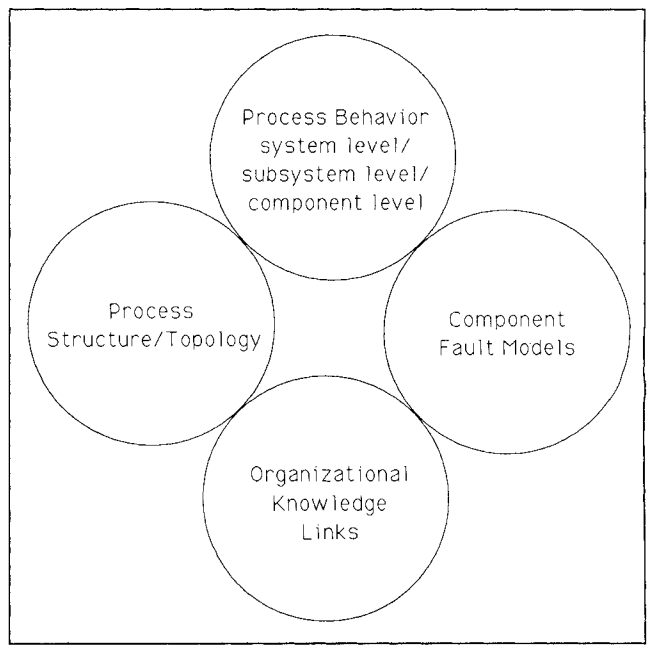

Figure 9. Knowledge requirements for DFS.

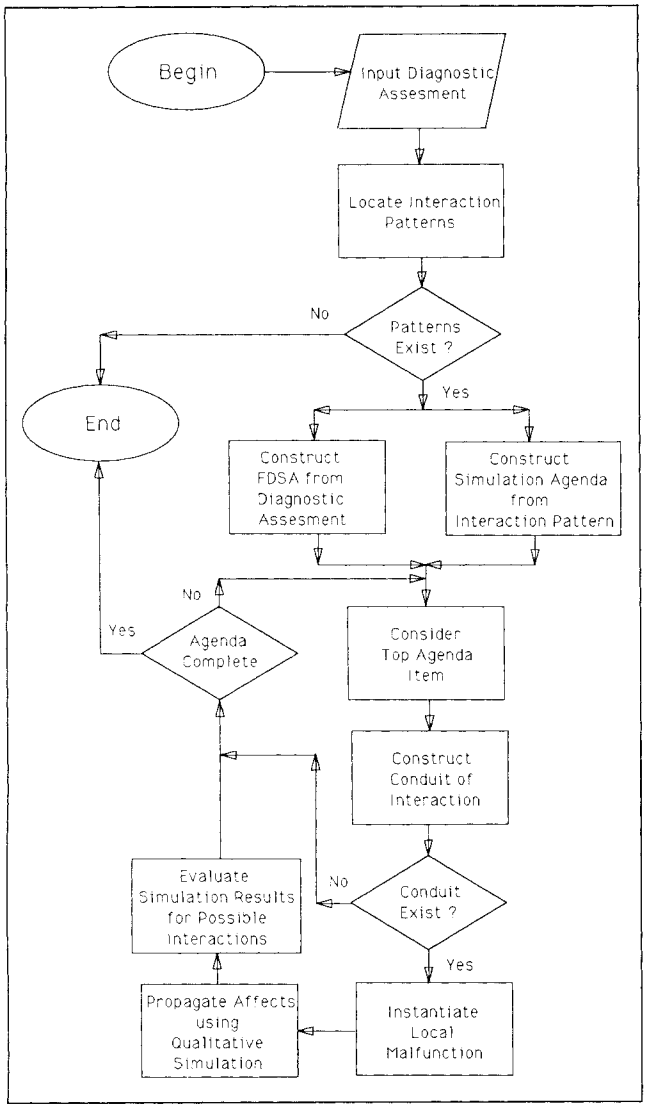

Figure 10. DFS inference procedure.

sults of the initial diagnostic assessment might look like the confidence-value hierarchy provided in Figure 8. In the event that possible interacting malfunctions exist, DFS is invoked in an auxiliary manner.

Overview of knowledge and inference in DFS

Figure 9 shows various forms of deep knowledge required for DFS. First, if DFS is going to reason about malfunctions, some form of knowledge concerning fault models must be available. Component fault models provide knowledge about how specific malfunctions affect local process variables. Second, knowledge about process structure is needed. For example, in the chemical process domain, knowledge about process structure is found usually in the process flowsheet. Knowledge about behavior must also be available at the system, subsystem, and component level. Knowledge about the structure and behavior will make it possible to reason about the cause-and-effect behavior across the system. Each of these forms of knowledge was used in one or more of the integration approaches discussed previously.

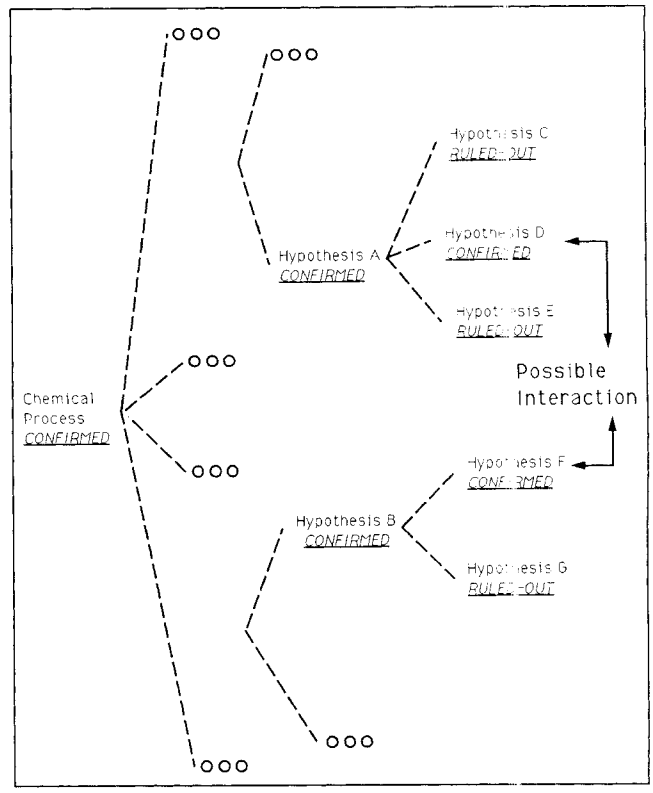

Figure 11. Interaction pattern one.

However, DFS requires an additional type of knowledge not found in other approaches. This additional requirement relates to organizational knowledge links from the compiled problem-solving architecture to the system structure. Hypotheses in the compiled problem solver map into relevant corresponding portions of the system. This knowledge serves to focus the causal reasoning on situation-specific portions of the process.

A flowchart of the DFS inference procedure that operates on the knowledge representations described above is shown in Figure 10. The diagnostic results are composed of the portion of the malfunction hierarchy explored during diagnosis, each malfunction hypothesis considered is given a confidence-value score (Figure 8). This map is searched for certain confidence-value patterns. These patterns identify specific situations of interaction and define a simulation agenda. Additionally, the map of hypotheses is transformed into a functionally decomposed structural abstraction (FDSA) of the system topology. The FDSA is a representation of the system structure at multiple levels of detail, appropriate for the diagnostic situation, and is constructed from the organizational knowledge links described in the knowledge requirements for DFS. The FDSA focuses in detail on the malfunctions and maintains higher-level abstractions for other portions of the system. According to the diagnostic goals, the FDSA is used to establish possible conduits for malfunction interactions. If there are no conduits, then simulation of the malfunction effects is not needed. If a conduit does exist, then candidate malfunctions are considered by propagating their local effects across the FDSA using qualitative simulation. The global effects on other malfunction hypotheses are evaluated by structured pattern matching on the simulation results. In essence, the qualitative simulation attempts to reenact the possible malfunction interaction.

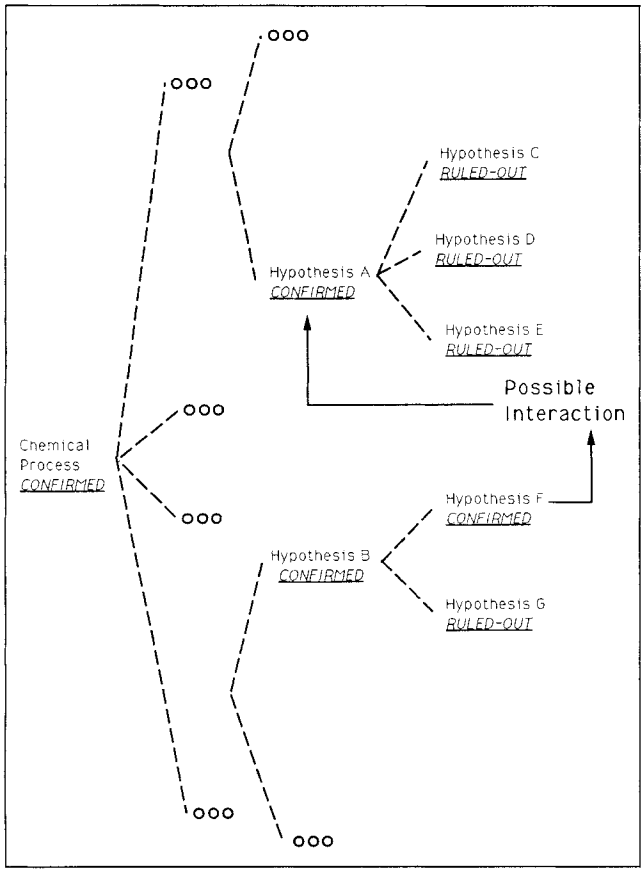

Figure 12. Interaction pattern two.

Confidence-value patterns

Possible malfunction interaction situations are identified by analyzing the results of the compiled diagnostic problem solver. As stated earlier, the results are in the form of a confidence-value hierarchy like that of Figure 8. Specific patterns of confidence values in the confidence-value hierarchy suggest possible malfunction interactions. Identifying the type of interaction specifies the diagnostic goals for using qualitative simulation and defines a simulation agenda for DFS. The confidence-value hierarchy makes it possible to identify possible malfunction interactions.

The current study has been restricted to two confidence-value patterns that have been defined. The first pattern is shown in Figure 11. The malfunctions represented by hypotheses C, D, E, F, and G are tip nodes. These nodes represent specific equipment malfunctions or incorrect process settings. Their respective parents, A and B, are not siblings, implying that they are in different functional sections of the process. Hypotheses D and F have been confirmed as malfunctions, and hypotheses C, E, and G have been ruled out. The question concerning this pattern is whether malfunctions D and F are causally related. The possible paths of interaction between D and F are not represented clearly by the hierarchy. Does D cause F? Does F cause D? Are they causally independent? These are the questions that can be raised. The simulation agenda for DFS in this interaction case would include two simulations: 1. cause malfunction D and determine if it affects F; 2. cause malfunction F and determine if it affects D. The interaction pattern in Figure 11 is realized in a case where a malfunction in the temperature regulatory system of the FCC fractionator causes hot spots in the feed preheat furnace (Ramesh et al., 1989).

A second confidence-value pattern is shown in Figure 12. Only a single malfunction has been confirmed (node F). A cluster of tip nodes has been ruled out, all of which are children of hypothesis A. With their immediate parent being confirmed (A), it is expected that one of the child tip nodes (C, D, or E) would be confirmed and not be ruled out. There are several possible reasons for this pattern. Of interest is a case of a secondary malfunction. The confirmed malfunction F is causing the parent malfunction abstraction A to be confirmed even though it is not part of the functional decomposition of A. F can influence A across the process topology without being a child of A. A secondary causal relationship may exist between F and A. A strict functional decomposition would not expose this relationship. The goal of DFS is to identify this secondary causal relationship. The simulation agenda involves a single simulation: cause malfunction F and determine if it affects A. Such secondary malfunctions suggested in Figure 12, were found in a diagnostic application for a terephthalic acid process (Shum et al., 1988).

Functionally-decomposed structural abstractions

To explore possible interactions among malfunction hypotheses, the system structure and behavior like that in Figure 1 must be considered. Figure 1 represents the chemical process at the lowest level of abstraction and the greatest level of detail required for the process of interest. However, more knowledge is available at this point in the problem solving concerning the chemical process. Not only a diagnostic assessment in the form of the confidence-value hierarchy is available, but the compiled problem solver identified the process subsystems that correctly function as well as the malfunction(s). The subsystems that correctly operate need not be analyzed to the detail of their process units. Instead, they can be represented as a consolidated whole.

Consider the diagnostic assessment in Figure 8 of the chemical process in Figure 1. Control valve 2 has been confirmed as a malfunction. Using organizational knowledge links from the compiled problem solver to the process structure, a case-specific view of the process is possible. This case-specific view of the chemical process is called the FDSA. Figure 13 provides the FDSA of Figure 1 given the diagnostic assessment in Figure 8. Figure 13 is the process at a level of detail appropriate for the diagnostic situation. When considering the interactions among multiple malfunctions, the FDSA can focus the detail at the appropriate level specific to the diagnostic results in the confidence-value hierarchy.

Before simulation takes place, some additional problem solving can be done. For the malfunctions to interact causally, there must be a conduit between them. Before simulation proceeds, the existence of such a conduit should be established. The process topology of that in Figure 1 can be used for this task, i.e., considering the process at the greatest level of detail. The FDSA is also appropriate for this task. The candidate hypotheses are represented as well as the structural connections. Since the FDSA contains abstractions of the process, it can be expected to have fewer connections and thus be easier to search. If no conduit between the malfunctions exist, then qualitative simulation can be avoided.

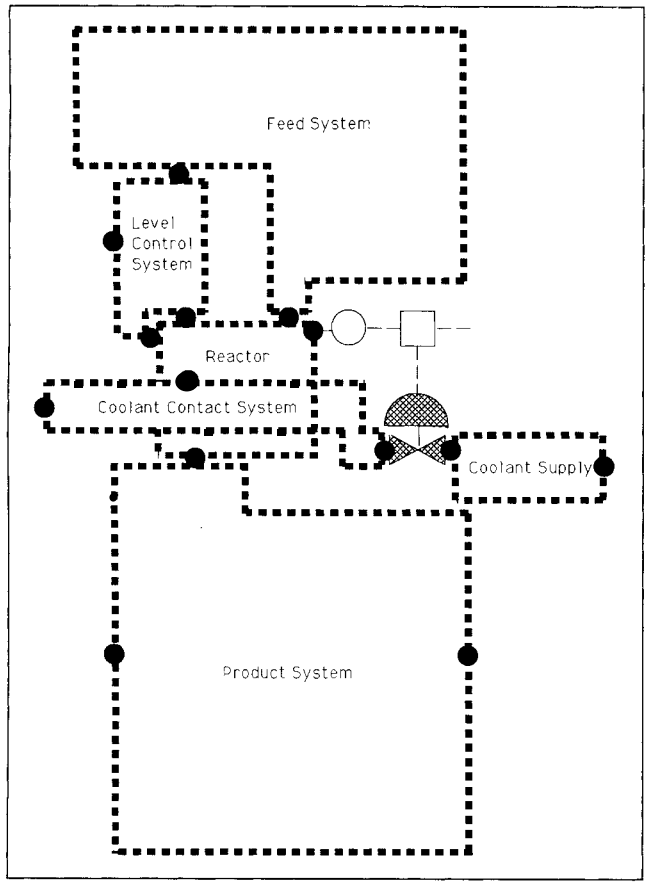

Figure 13. FDSA of the chemical process.

Component fault models

To determine if malfunction A causes malfunction B, first it is necessary to have knowledge about the local effects of malfunction A. Local effects refer to how a particular malfunction affects local process variables. For example, a leaking pipe will cause loss of material, a stuck control valve will eliminate control actions, and fouling causes a reduction in heat transfer. This knowledge must be enumerated for all the tip-level hypotheses in the diagnostic hierarchy. The effect on local process variables represents the initial conditions for qualitative simulation.

The use of fault models is found in many diagnostic systems: MODEX2 (Venkatasubramanian and Rich, 1988), MIDAS (Finch and Kramer, 1989), and MIMIC (Dvorak and Kuipers, 1989). A process equipment item can have several behaviors. A failure mode identifies a portion of process equipment behavior that is faulty and focuses on the constraint relationships that must be modified as part of this failure.

If no failure mode is identified for a malfunctioning equipment item, then all the constraints describing the unit's behavior are suspended. This approach of constraint suspension has been used successfully in electronic diagnosis (Davis, 1984).

Qualitative simulation

Once the FDSA is constructed, the relevant malfunction is identified, and its effect on local process variables is enforced, the procedure becomes one of qualitative simulations. The qualitative simulation propagates the effect of a malfunction across the FDSA. For this to occur, the FDSA must map into the appropriate qualitative modeling primitives. From the examination of other approaches to using deep knowledge and reasoning, there are several ways to qualitatively model processes and process behaviors. The FDSA as a representation, however, is independent of modeling approach. Prior to simulation, the FDSA maps into the appropriate modeling primitives creating a qualitative model specific to the diagnostic case. The important issue is that the modeling approach must be robust enough to represent the engineering systems of interest. It must also be flexible enough to represent processes at several levels of detail.

A major concern and criticism of qualitative simulation are the generation of multiple and incorrect process behaviors (Kramer and Oyeleye, 1988). The source of this issue is the qualitative nature of the processing. The simulation at some level is necessarily vague and imprecise. Several efforts have been devoted to dealing with this issue. Kuipers' (1986) constraint ontology is mathematically rigorous enough to guarantee the generation of at least the correct result, when the qualitative model is derived from a solvable system of ordinary differential equations. Attempts have been made to use specific quantitative information to constrain the branching during qualitative simulation (Kuipers and Berleant, 1988) as well as extending the qualitative mathematics to higher-order derivatives (Kuipers and Chiu, 1987). In the case of the Forbus' modeling approach, efforts have been devised to represent process abstractions at several levels of detail (Faulkenhainer and Forbus, 1988).

DFS addresses this multiplicity situation in two ways. The FDSA is a diagnostically derived process abstraction. Using the FDSA as the basis for the qualitative physics model avoids unnecessary process details that can contribute to the generation of multiple results during qualitative simulation. In this sense it parallels many of the issues described by Faulkenhainer and Forbus (1988); however, their work does not address abstraction from a diagnostic foundation. Kramer and Oyeleye (1988) identified the need to correctly locate functioning portions of the process, especially control loops to reduce multiplicity. In the case of DFS, part of the compiled diagnostic problem solver's operation is to identify portions of the plant that correctly operate as well as identify malfunctions.

Pattern matching and evaluation

Once the qualitative simulation is complete, the diagnostic goals are evaluated. The evaluation involves the pattern matching of specific hypotheses from the original compiled problem solver against the results of the qualitative simulation. If any multiplicity of final results occurs, then each individual result is evaluated. The evaluation phase might involve using the SPM knowledge of a single-tip hypothesis or the SPM knowledge of the tip hypothesis and all its ancestors. The case will depend on the knowledge engineering of the SPM knowledge and relationship among related hypotheses. The results of the evaluation depend on the type of interaction suspected, which is derived from the confidence-value patterns in Figures 11 and 12.

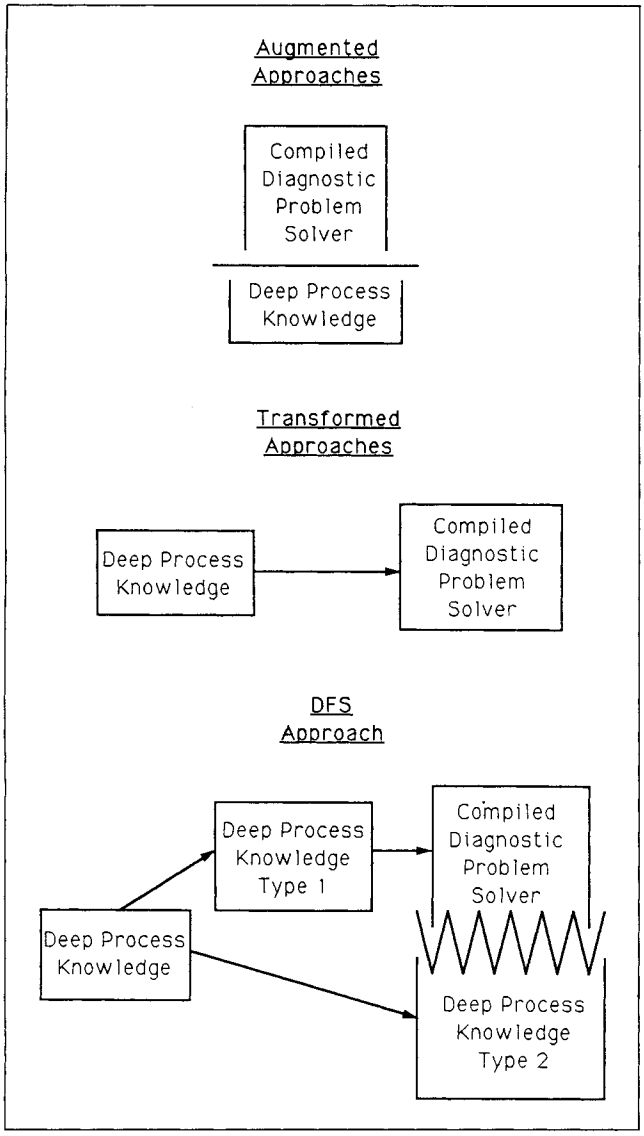

Figure 14. Approaches to integration.

Comparison with previous approaches

DFS makes a very different commitment to integration than augmented or transformed approaches. In a sense, DFS is a synthesis of the two approaches to integration. Figure 14 demonstrates the differences in integration approaches. The DFS classifies deep process knowledge into two types. Type 1 knowledge serves as a knowledge source for the compiled diagnostic problem solver as seen in the transformed approaches. In the discussion of generic tasks, it was pointed out that deep knowledge can be a source for HC and SPM. Type 2 knowledge is the deep knowledge used at run-time. Though knowledge is separated much like the two-tier approaches, the integration in DFS is tightly woven and not stratified. In DFS, the compiled problem solver has links with the deep knowledge at several levels. The results of the compiled problem solver strongly impact what deep knowledge is used in DFS. Like transformed approaches malfunctions are simulated, but in the case of DFS the simulations are used only on an as-needed basis to resolve the cases of interacting multiple malfunctions. DFS's basis in the task architecture allows this novel and tightly integrated approach to using deep knowledge.

Conclusions

The DFS problem-solving task was designed to resolve multiple interacting malfunctions using deep knowledge. This task uses qualitative simulation in a diagnostically focused manner for the resolution of interacting multiple malfunctions. The close integration with other problem solvers represents an evolutionary approach to using qualitative simulation in diagnosis. Under ordinary conditions, qualitative simulation can be a computationally explosive procedure. Because it is part of a distributed problem-solving architecture, DFS uses qualitative simulation in a constrained and diagnostically specific manner. The results of the classificatory problem solver suggest the existence of specific interacting malfunction scenarios. In addition, a situation-specific view of the process is assembled, providing the appropriate level of process detail. Qualitative simulation is performed only if a conduit of interaction exists between the malfunctions. As an integrated approach, DFS brings together multiple sources of knowledge, a situation-specific interpretation of diagnostic results and a balance between the use of run-time simulation and compiled problem solving in the activity of diagnosis.

Literature Cited

Bylander, T., "Some Causal Models are Deeper than Others," *Artif. Intell. in Medicine*, **2**(3) (1990).

Calandranis, J., G. Stephanopoulos, and S. Nunokawa, "DiAD-Kit/Boiler: On-Line Performance Monitoring and Diagnosis," *Chem. Eng. Prog.*, **68**(1), 60 (1990).

Chandrasekaran, B., "Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design," *IEEE Expert*, **1**(3), 23 (1986).

Chandrasekaran, B., "Towards a Functional Architecture for Intelligence Based on Generic Information Processing Tasks," *Proc. Int. J. Conf. on AI*, Milan, Italy (Aug., 1987).

Chandrasekaran, B., "Toward a Taxonomy of Problem Solving Types," *AI Mag.*, **4**(1) (1983).

Chester, D. L., D. E. Lamb, and P. S. Dhurjati, "An Expert System Approach to On-line Alarm Analysis in Power and Process Plants," *ASME-Comp. in Eng.*, **1**, 345 (1984).

Clancey, W. J., "Viewing Knowledge-Bases as Qualitative Models," *IEEE Expert*, **4**(2), 9 (1988).

Davis, J. F., W. F. Punch III, S. K. Shum, and B. Chandrasekaran, "Application of Knowledge-Based Systems for the Diagnosis of Operating Problems," AIChE Nat. Meeting, Chicago (1985).

Davis, R., "Diagnostic Reasoning Based on Structure and Behavior," *Artif. Intell.*, **24**, 348 (1984).

deKleer, J., and J. S. Brown, "A Qualitative Physics Based on Confluences," *Artif. Intell.*, **24**, 7 (1984).

Dvorak, D., and B. Kuipers, "Model-Based Monitoring of Dynamic Systems," *Proc. Int. Joint Conf. on AI*, Detroit (1989).

Faulkenhainer, B., and K. D. Forbus, "The Organization of Large Scale Qualitative Models," *Proc. Nat. Conf. on AI*, Morgan Kaufmann, Los Altos, CA (1988).

Finch, F. E., and M. A. Kramer, "The Handling of Dynamics, Multiple Faults, and Out-of-Order Alarms," AIChE Nat. Meeting, Houston (Apr., 1989).

Finch, F. E., and M. A. Kramer, "Narrowing Diagnostic Focus Using Functional Decomposition," *AIChE J.*, **34**(1), 25 (1987).

Fink, P. K., and J. C. Lusth, "Expert Systems and Diagnostic Expertise in the Mechanical and Electrical Domains," *IEEE Trans. on Systems, Man, and Cybernetics*, **SMC-17**(3), 340 (1987).

Forbus, K. D., "Qualitative Process Theory," *Artif. Intell.*, **24**, 85 (1984).

Grantham, S., and L. Ungar, "A Qualitative Physics Approach to

Troubleshooting Chemical Plants," AIChE Nat. Meeting, San Francisco (Nov., 1989).
Kramer, M., and O. O. Oyeleye, "Qualitative Simulation in Chemical Engineering: Steady State Analysis," *AIChE J.*, **34**(9), 1441 (1988).
Kuipers, B., "Qualitative Simulation," *Artif. Intell.*, **29**, 289 (1986).
Kuipers, B., and D. Berleant, "Using Incomplete Quantitative Knowledge in Qualitative Reasoning," *Proc. Nat. Conf. on AI*, Morgan Kaufmann, Los Altos, CA (1988).
Kuipers, B., and C. Chiu, "Taming Intractable Branching in Qualitative Simulation," *Proc. Int. Joint Conf. on AI*, Morgan Kaufmann, Los Altos, CA (1987).
McDowell, J. K. J. F. Davis, M. A. Abd-Allah, and M. Aref, "A Comparison of Compiled Reasoning Systems and Model-Based Systems and Their Applicability to the Diagnosis of Avionics Systems," *Proc. IEEE Nat. Aerospace and Electronics Conf.*, Dayton, OH (1990).
McDowell, J. K., D. R. Myers, and J. F. Davis, "A Changing Perspective on Model-based versus Compiled Knowledge Systems: an Examination in Terms of a Fielded Expert System," Workshop on AI in Manufacturing, *Int. Conf. on AI*, Detroit (1989).
Myers, D. R., J. F. Davis, and C. E. Hurley, "Application of Artificial Intelligence to Malfunction Diagnosis of Sequential Operations Which Involve Programmable Logic Controllers," AIChE Nat. Meeting, Houston (Apr., 1989).
Myers, D. R., J. F. Davis, and C. E. Hurley, "An Expert System for Diagnosis of a Sequential, PLC-Controlled Operation," *Artificial Intelligence Applications in Process Engineering*, M. Mavrovouniotis, ed., Academic Press, New York (1990).
Oyeleye, O. O., F. E. Finch, and M. A. Kramer, "Qualitative Modeling and Fault Diagnosis of Dynamic Processes by MIDAS," AIChE Nat. Meeting, San Francisco (Nov., 1989).
Petti, T. F., J. Klein, and P. S. Dhurjati, "Diagnostic Model Processor: Using Deep Knowledge for Process Fault Diagnosis," *AIChE J.*, **36**(4), 565 (1990).
Ramesh, T. S., J. F. Davis, and G. M. Schwenzer, "CATCRACKER: An Expert System for Process and Malfunction Diagnosis in Fluid Catalytic Cracking Units," AIChE Nat. Meeting, San Francisco (Nov., 1989).
Ramesh, T. S., S. K. Shum, and J. F. Davis, "A Structured Framework for Efficient Problem Solving in Diagnostic Expert Systems," *Comput. Chem. Eng.*, **12**(9/10), 891 (1988).
Shum, S. K., J. F. Davis, W. F. Punch III, and B. Chandrasekaran, "An Expert System Approach to Malfunction Diagnosis in Chemical Plants," *Comput. Chem. Eng.*, **12**(1), 27 (1988).
Shum, S. K., M. S. Gandikota, T. S. Ramesh, J. K. McDowell, D. R. Myers, J. R. Whiteley, and J. F. Davis, "A Knowledge-Based System Framework for Diagnosis in Process Plants," *Proc. Power Plant Dynamics, Control and Testing Symp.* (1989).
Steels, L., "Components of Expertise," *AI Mag.*, **11**(2), 28 (1990).
Venkatasubramanian, V., and S. H. Rich, "An Object Oriented Two-Tier Architecture for Integrating Compiled and Deep-Level Knowledge for Process Diagnosis," *Comput. Chem. Eng.*, **12**(9/10), 903 (1988).

Manuscript received Sept. 13, 1990, and revision received Feb. 6, 1991.