DRA

REPORT T-135 DECEMBER 1983

CSL COORDINATED SCIENCE LABORATORY

MULTILEVEL SEMANTIC ANALYSIS AND PROBLEM-SOLVING IN THE FLIGHT DOMAIN

FINAL REPORT

NASA GRANT NAG-1-288

JULY 11, 1982 - SEPTEMBER 30, 1983

R.T. CHIEN
D.C. CHEN
W.P-C. HO
Y.C. PAN

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

"Multilevel Semantic Analysis and Problem-Solving in the Flight Domain"

Final Report

NASA Grant NAG 1-286

July 11, 1982 - September 30, 1983

Submitted by

Professor R. T. Chien
Principal Investigator

Prepared by

R. T. Chien
D. Chen
W. Ho
Y. Pan

TABLE OF CONTENTS

1. HIERARCHICAL PLANNING AND MONITORING WITH CONCEPTUAL LEVELS

This work deals with the use of knowledge-base architecture and planning control mechanisms to perform an intelligent monitoring task in the flight domain. Progress made this past year centers on the final refinement of the conceptual levels planning approach and the implementation of this design. At this time the implementation is nearly completed. The route level, the trajectory level, and parts of the aerodynamics level can now be demonstrated.

The conceptual levels approach proposes an alternate method of obtaining global viewpoint: model the domain at different degree of global viewpoint. A complex real-world domain such as the flight domain requires global viewpoint during planning. Let us define the primitive ground-level operators to be the physical control actions the flght crew perform during flight. Suppose we record the flight crew during flight with a video recorder. A physical action is what the flight crew does each time a control element is changed. Since the flight domain is a dynamic domain and some of these control elements need to be adjusted frequently one can see that the state space is quite large. To plan a flight at the ground-level, worrying about the control positions second by second, is clearly not fesible.

In addition to being a complex domain, the flight domain places additional demand on the planner because the plan must conform to domain constraints. Some constraints are global while others are more local. For example, to have a safe flight, the aircraft should not run out of fuel, should fly a navigable course such that it does not get lost, should not fly into a mountain nor through icing condition nor turbulence, should not fly so fast that the airframe is damaged nor so slow that the aircraft stalls. The planner must generate a plan that achieves the goal while satisfying these constraints.

It is interesting to note that these constraints have different scope. Scope means the extent of coverage over the ground-level plan, roughly analogous to global viewpoint. A constraint such as "the aircraft should not run out of fuel" has very large scope since it affect the structure of the entire plan. A constraint such as "do not stall the aircraft" has very small scope since it affects very small portion of the ground-level plan. A constraint such as "do not fly in turbulence" has a scope somewhere in between. The reader may also notice that these constraints are not directly related to each other. For example, the "do not run out of fuel" constraint, while operating at a larger scope, is not a more general version of the "do not fly in turbulence" constraint.

The immediate effect of the conceptual levels approach is to break the complex domain into simpler homogeneous minidomains. By specifying the minidomains along different facets of the domain, the domain complexity is reduced to several nearly independent less complex domains which are causally complete by themselves. This reduction of complexity is similar to the problem-solving paradigm that if a task is too difficult, divide it into easier subtasks and solve the subtasks. The division here is to partition the domain semantics along the aspects of the domain, thus reducing the search space inside a subdivision. Instead of complexity reduction through task decomposition, the complexity reduction is through domain knowledge partitioning.

While domain complexity reduction is desirable, the flight planner requires global viewpoint. Global viewpoint can be obtained through vertical domain knowledge decomposition. This is done by constructing minidomains of different scope, or in other words, model the domain at different degree of global viewpoint. It is vertical because the minidomain of the largest scope makes decisions which guide planning at the minidomain of lesser scope.

Domain knowledge decomposition achieves a reduction in complexity, but with the gain also comes the drawback of limited viewpoint. The cause of this limited viewpoint is the partitioning of the domain knowledge. The conceptual levels architecture achieves both broad viewpoint and narrow, focused viewpoint because of the multiple conceptual levels. The top level is constructed to support the broad viewpoint and the bottom level is constructed to support the tight detailed viewpoint. The problem is that since not one conceptual level can cover both the broad and tight viewpoint, the planner cannot see the total picture at a given time. Thus the planner cannot plan with absolute accuracy at the high level, nor can it appreciate its role at the low level.

Since the levels hierarchy is designed to provide global viewpoint for the low-level planners, the inability to look ahead at low level is not a problem. However, the high-level planner, not able to plan with absolute accuracy, is a problem. Planning proceeds top-down; the route planner plans to satisfy global constraints, then the trajectory planner plans to satisfy constraints of intermediate scope, and finally the aerodynamics planner flies the aircraft and tells the subsystems level what is necessary to fly. The difficulty is that since the route planner is buffered from the engine system and the actual flying, it does not know for certain what are the aircraft range and ceiling. And yet, if the route planner consider the trajectory, the aerodynamics control settings, and the subsystem control settings to determine the exact range and ceiling values while considering traversing an airway segment, the route planner would be bogged down in detail which the architecture is designed specifically to avoid.

This is an insoluable catch-22 problem inherent to this architecture. One must choose between risking the combinatorial explosion to ensure an accurate world model at the high level or risking a bad plan at the high level due to an inaccurate world model. The least of two evils appears to be the second choice: plan at high level with a possibly inaccurate world model. This approach introduces two new wrinkles to top-down planning: replanning may be necessary and bottom-up verification of the world model is required.

Planning is necessarily top-down in this architecture because the low-level planner cannot make global decisions and yet it cannot plan without the global decisions since these global decisions influence its planning. Planning decisions are made at high level based on a possibly inaccurate world model, and yet this high-level world model can not be verified until the low-level plan is set. When the low-level plan is done, it is examined to verify that it agrees with the model of the level above, and agian with the levels above. The plan is not finished until all the models agree.

Model updating is also necessary when the external world changes unexpectedly. For example, suppose the flap is stuck after takeoff. The trajectory level and the route level should be aware of this and check their plans to make sure they are still good. Similarly, a failed engine or an unexpectedly stiff headwind requires model updating and plan checking. The model updating is initiated at the level where the change occurred. For example, a stuck flap is initiated at the aerodynamics level, and a fuel leak is initiated at the subsystems level. The extent of model updating depends on the situation and is controlled by inter-level planning control.

The conceptual levels planner is implemented in Franz LISP. More specifically, it is implemented in rule system and frame system. All static knowledge structures are implemented in a locally developed frame system, written in Franz LISP. All active knowledge structures are implemented in a rules system, more specifically, the OPS5 rules system written in Franz LISP. Both knowledge structure systems are contained in the same lisp environment running on a VAX 11/780 computer. In addition, a PDP-11/45 computer and a Ramtek color graphics system are used to display the aircraft environment.

The rules contain the meta-planning knowledge. This meta-planning knowledge contain both the intra-level planning control knowledge and the inter-level planning control knowledge. Presently, approximately 100 rules are implemented. These rules control the planning at the route level, the trajectory level, and the partially implemented aerodynamics level. As the levels are further developed and refined, there may be approximately 200 rules.

The frame system is used to represent domain knowledge and the plan structure. In fact, since there are four levels, the entire plan structure is made up of four substructures. The domain knowledge represented in a frame consists of knowledge such as the airports, the vortac transmitters, the airway segments between the vortac transmitters, the attainable cruise trajectories, the attainable climb trajectories, the attainable descent trajectories, the cruise phase, the climb phase, the descent phase, the climb force vector equilibrium system, the cruise force vector equilibrium system, etc. The plan operators are represented in frame structure, and naturally the resulting plan is similarly represented.

The planning system is presently implemented down to the aerodynamics level. The planning process is displayed both on a terminal and on the Ramtek graphics monitor. The aerodynamics level will be completed in the near-term future. More detailed discussion of the theory and the implementation is available shortly in a technical report.

2. FUNCTION-DIRECTED MECHANISM RATIONALIZATION

2.1. Scenario

Consider the following scenario. An expert in a specific mechanism domain is given a diagramatic representation of the physical structure of an instance of a specific mechanism. Furthermore, let us assume that he is told the name of the mechansim. This name is meaningful to him - he knows what the abstract function of the mechanism instance must be. What we require of this expert is an explanation of the instance behavior of the instance in question. This explanation is a product of the application of the expert's mechanism understanding process.

2.2. Mechanism Understanding

We are interested in the underlying concepts and principles of this mechanism understanding process.

There may be several instances of a specific mechanism - for example, there are several instances of the amplifier which use the transistor as the active element, and several instances of the amplifier which use the operational amplifier as the active element. As a further example, there are several instances of the thermostat - one incorporating a bimetallic strip as the temperature-sensitive element, and another incorporating a gas bellow as the temperature-sensitive element. Thus the physical structures of the various instances in question may appear quite different. However, the abstract function of these instances remains the same. The instance behaviors, which are generated analytically from the physical structures, will accordingly be unique to the instance in question. However, the expert seems to be able to cope - he can rationalize an instance of a mechanism which appears familiar, but which he has never seen before. Accordingly, the generation of an explanation of the instance behavior must begin from the known abstract function of the mechanism instance in question.

Generally speaking, the understanding process is a forward qualitative reasoning or qualitative simulation. The explanation of the instance behavior of the instance in question is an artifact of the expert's analytical understanding process. This understanding process may be based on functional component state models as the qualitative knowledge primitives [De Kleer], or it may be based on functional process models as the qualitative knowledge primitives [Forbus]. Some input signal is introduced as the input of the instance or situation. This input signal is then propagated through the qualitative constraints imposed by the primitive models, and the qualitative constraints imposed by their connection scheme into an instance or situation. The imprecise resolution of this analysis is introduced because of the approximate nature of qualitative simulation. Thus, qualitative simulation leads to many possible interpretations of instance behavior, each with a unique set of key assumptions chosen at decision points encountered during the course of that simulation. We are identifying such ambiguity classes.

We believe that a qualitative simulation process of some kind is necessary for understanding how an instance of a mechanism can achieve the characteristic function of that mechanism.

This qualitative simulation step should not be applied without some careful, knowledge-based focusing and simplification of the instance in question. Domain knowledge of the abstract mechanism function and domain knowledge of physical substructures encountered in the past can and should be utilized to properly frame the mechanism instance for understanding through qualitative simulation.

2.3. Problem Focus

The understanding process begins with the expert knowledge of the abstract function of the mechanism. The expert knows what the instance must be designed to achieve. This expert knowledge serves as global control to direct the analytical understanding process. It is used to direct a decomposition of the understanding problem into many separate, almost-independent, stand-alone subproblems. In doing so, this expert knowledge is used to constrain the understanding analysis by establishing all of the environments of complete definitive mechanism operation, one for each subproblem. An environment is identified with a functional goal that the instance behavior must achieve. The composition of all of the functional goals of all of these almost- independent subproblems is the overall mechanism function. An environment is specified by adding the knowledge of what input signal is characteristic of the subfunction goal, and what output signal is expected to be achieved as an effect of the constraints imposed by the components and connection scheme of the instance in question.

Take as an example, the nand circuit. The abstract function of the nand circuit is a multi-state truth table description using the language of logical '0' and '1'. Knowing that the instance in question is a nand circuit, the appropriate chunk of knowledge comes to the fore. The expert immediately makes many knowledge based assumptions. Among these assumptions, he decomposes the understanding of the instance into four almost-independent subproblem environments - one for each situation associated with a truth table entry. The composition of the resulting explanations of all of these understanding subproblems is a complete explanation of the overall nand circuit function. Associated with each environment is the function goal of that environment - achieving the appropriate logic signal vector at the output terminals, when the characteristic input logic signal vector is introduced at the input terminals. The expert thus decomposes his analytic understanding problem into four separate perspectives (a perspective is a physical structure in an environment with a specific function goal). These four perspectives can be reduced to three if symmetry of nand circuit input terminals is taken into account - the (0 1 -> 1) situation and the (1 0 -> 1) situation are symmetrical.

Problem focus is achieved by using the abstract function of the instance in question to direct the decomposition of the understanding problem. A perspective allows us to focus on the subproblem at hand to the exclusion of all other subproblems. For each subproblem, it identifies the specific environment in which that instance must operate and the specific function goal or subfunction that that instance behavior must achieve. A perspective also focuses the subproblem at hand by telling us what irrelevant details to ignore and discard in the analytical understanding process for that subproblem. For each subproblem of the nand circuit, we are no longer concerned with the specific node voltage values. We instead speak in a language of logical '0' and '1'. The perspective also dictates the type of qualitative analysis which should be used to analyze each particular subproblem. We know not to perform small signal perturbation analysis to determine the relevant instance behavior. We are no longer interested in the state transition history of the multi-state components. We instead know to perform steady-state dc analysis for each nand circuit perspective.

2.4. Problem Simplification

The analytical understanding process is controlled and organized in a top-down manner into perspectives as directed by the mechanism function. The understanding problem is thus focused through a decomposition process. A process to simplify the subproblem at hand is the recognition and consequent clustering of physical substructures into single behavioral entities. This process works in a bottom-up manner since it begins from the physical structure of the instance in question. The expert often distinguishes and circles substructures on the diagramatical represenation of the instance in question. He recognizes these substructures as ones that he has seen before. This recognition simplifies the physical structure of the instance in question by a physical abstraction of detail. Each recognized substructure is composed in a sense and replaced by a single macro-component with unit function. The result is a simpler physical structure with fewer constituent components in a simpler connective topology.

Take as an example, the two-stage dc amplifier. The abstract function of the two-stage dc amplifier is an equational description using the language of 'Gain' and 'Offset'. An amplifier expert would immediately recognize that the instance contains two physical substructures - amplifier stages - which each contain three components, and one physical substructure - voltage divider - which contains two components. As a result, the expert can generate another level of diagramatic representation of the physical structure of the mechanism instance. This new representation level is the result of an abstraction of the initial diagrammatic representation of the physical structure of the mechanism instance. He composes these three physical substructures into three macro-components through a physical abstraction of detail. The result is a simpler physical structure of three constituent components in a simpler linear connective topology. Associated with each macrocomponent is its own abstract function description. Each amplifier stage has its own component parameter 'Gain'. Thus, there is an associated abstraction of the parametric description of the new components in the new level of physical structure. The physical structure may be reduced even further, to two components, if the expert recognizes that the two amplifier stages can be composed into a single functional composite amplifier stage - a still more abstract level of physical stucture. The gain parameters of the two amplifier stages are multiplied together to produce the more abstract parameter gain of the composite amplifier stage.

Problem simplification is achieved by using the knowledge gained from past experience to recognize and abstract familiar substructures ensconced in the mechanism instance in question. Each macrocomponent model is described or encoded using syntactically the same format as that used to describe primitive component models. In this way, a physical structure composition hierarchy is produced that enables explanation at various levels of detail. The analysis is also focused by working with the more abstract parameters which describe the macro-component function rather than the more numerous, less abstract parameters which describe the functions of the substructure components. Physical abstraction focuses on the appropriate level of the parameter language to be used in the description of instance behavior. Unlike perspectives, the introduction of levels of detail summarizes and saves detail, which may be recalled for consideration rather than being totally discarded.

3. USING DEEP-LEVEL MECHANISM MODELS FOR DIAGNOSES OF DEPENDENT FAILURES

3.1. Overview and Summary

This final report summarizes the result of our research on the model-based diagnoses under many years of NASA support. We first briefly overview our approach to the mechanism diagnosis problems. In our diagnosis methodology, we decompose the diagnosis task into two subtasks: **hypothesization** and **verification.** In the hypothesization phase, a set of failure possibilities is heuristically postulated. Then in the verification phase, the expected behavior of each failure hypothesis, as predicted from reasoning with the deep-level mechanism model, is matched against the symptoms actually observed. The main theoretical issues of our research lie in the development of a deep-level mechanism model (which we call the **state-transition model**, and the use of such a model for reasoning about the qualitative behavior of a faulty mechanism (a process which we call the **predictive analysis**). Those theoretical issues are thoroughly discussed in Pan's Ph.D. dissertation [1], which also investigates two interesting aspects of hypothesis verification, namely, the use of **qualitative symptom features**, and the use of **transient symptoms** in verification.

In this report, we focus our discussion on issues related to the hypothesization process which, though less theoretical in its nature, serves as an important link between the proposed diagnosis theory and its real-world applications. In the next section, we first use an idealized mechanism model (called the **Production-Line**

model) to explain: (1) how traditional diagnosis approaches have been "made" easy, and (2) what the underlying assumptions being imposed on these approaches are. Then we discuss our heuristical approach for dealing with a non-ideal situation: the real-world analog mechanisms. A **Flow-Causality model** is introduced to decompose the functionality of a mechanism into two levels: the Causal Level in which the diagnostic principles developed for PL-modeled mechanisms can be "heuristically" applied (with exceptional cases), and the Flow Level in which special (expert) diagnostic knowledge for each type of flow systems has to be adopted. As the result of the discussions on diagnoses based on the PL-mechanisms, the limitations of a straightforward diagnostic approach can be better understood, and the analyses of its weakness forms the basis for the future development of a diagnosis strategy applicable to general analog mechanisms.

At the end of this chapter, we propose a system organization for the future intelligent diagnosis system constructed based on our theory.

3.2. Generations of Failure Hypotheses

Within the context of a time-complicated mechanism with dependent-failure possibilities, the generation of failure hypotheses is a task of postulating possibilities of the **primary failure** based on the initial symptom-snapshot which triggers the detection of a failure. Our hypothesis generation is similar to traditional diagnosis problems in two ways: (1) the task is based on symptoms observed at one **single** snapshot, and (2) the goal is to assert a set of single-failure possibilities which can potentially causes the observed symptoms. In view of our overall diagnosis approach, the verification process will refine the result of this failure hypothesization to (1) incorporate the use of time-elapsed symptoms for resolutions among the hypotheses, and (2) understand and thus explain dependent failures.

A naive approach for generating failure hypotheses is to propose all possible failures. However, by doing so, an enormous computational problem will be created in the verification process. A smarter approach will adopt some heuristically-based reasoning process to select a set of most-likely failure possibilities among the space of all possible failures.

A straightforward approach is to adopt the production-system paradigm [2] by which expert's rules for diagnoses are encoded and utilized to map the symptoms to possible failures. While not questioning the effectiveness of the production-system approach in dealing with a specific mechanism, such approach totally bypasses the well-formulated modeling knowledge which engineers develop to analyze the domain. An alternative approach will be to develop a functional model of the mechanism, and to use the model in deriving failure hypotheses from observed symptoms. In comparing these two approaches, the model-based approach is theoretically more interesting in that the methodology we develop in dealing with the modeling of mechanisms and the using of models can be applied on other mechanisms, while with the production-system approach each mechanism has to be treated as a special case[1].

In this chapter, our focus is on the model-based approach. We first discuss the hypothesization problem in an idealized mechanism modeled by the "loopfree-production-line" model, which helps us understand the limitations of existing AI diagnosis works and sets the stage for the discussion of our approach in dealing with non-ideal situations. We then discuss the hypothesis generation problem in an analog mechanism with a proposed functional model, called the "flow-causality model".

3.2.1. The Loopfree-Production-Line Model for Ideal Mechanisms

To understand how failure possibilities can be hypothesized from symptoms by reasoning with a functional model of the mechanism, we first study an ideal situation in which such tasks can be performed with a straightforward diagnostic logic. Assuming that we are dealing with an ideal mechanism in which interactions among sub-modules can be modeled in such a way that (1) propagations of effects (signals) among modules is always "unidirectional", namely, whatever failure occurred beyond the output of a module will not affect the behavior of upstream modules, (2) a module will behave differently if its input is abnormal, and (3) there is no loop existing in propagations of effects among modules. For the purpose of easy reference, we will call such an idealized

[1] However, a diagnosis approach based on the production system paradigm has its advantage in that it can encode global diagnostic knowledge which is not direct rationalizable from deep-level mechanism models. Examples of such knowledge are special experiences (e.g., engine vibrates), priori-probabilities, etc.

mechanism the **loopfree-production-line model** (since a production line is usually loop-free, we call it a "production-line" (**PL**) model for short) for its similarity to the station-to-station interactions in the production-line of a factory. As shown in figure 3.1(a), a typical production line has a sequence of processing stations (A,B,C,D in this example). Interactions between two processing stations are done by a conveyer belt which carries partial products from one station to another. If a failure occurs (say in station C, as shown in figure 3.1(b)), we will observe symptoms in station C (partial products piling up) and stations downstream to C (which is station D with no "input" products), but never on its upstream stations A and B.

In a mechanism describable by the loopfree-production-line model, the partially-ordered causalities are implicitly established among its modules, corresponding to the directions of signal propagations. With the assumption that there is a way to determine the normal/abnormal status of an operational module[2], the isolation of the faulty module can be achieved by backtracing along the causal paths of the mechanism. In the following paragraphs, we discuss four basic diagnosis principles, (namely, causal-backtracing, fault-identification, common-cause, implied-exoneration), within hypothetical situations for their applications.

[2] In WATSON's task of isolating radio-receiver failures at the stage-level [3], the operational status of each stage is defined by the "normality" of stage-output signal.

ORIGINAL PAGE IS
OF POOR QUALITY

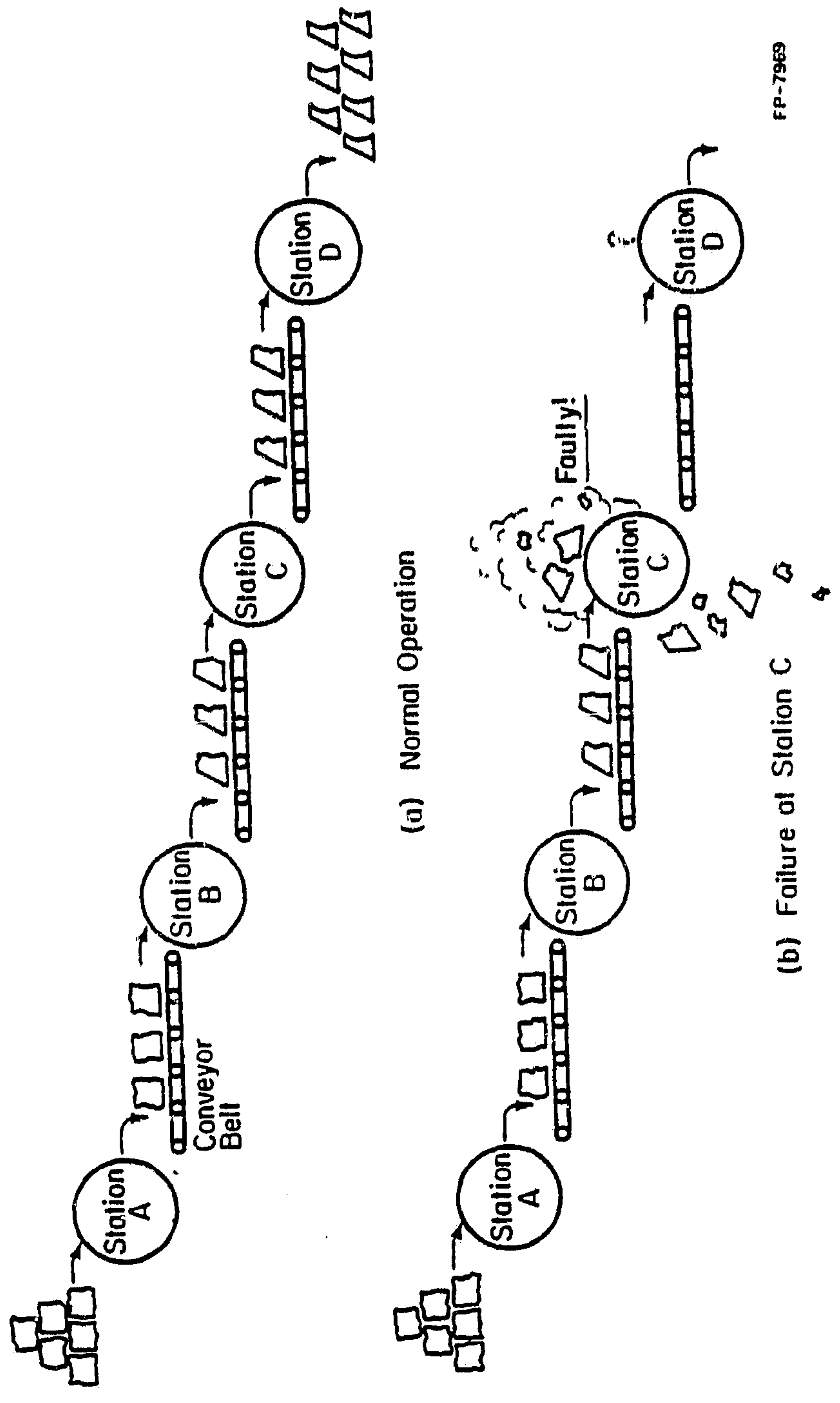


Figure 3.1 A Loopfree-Production-Line (PL) Model and a Failure Example

As shown in figure 3.2(a), interactions among functional module A, B, and C are such that A enables B and B enables C. Operational statuses of B and C are known to be abnormal while A is known to be normal. To isolate the possible failure in this example, we first reason that since C is enabled by B and B is abnormal, the the abnormality of C must be inherited from the abnormal B (by the "uni-directional" assumption), and thus C is exonerated (by the "single-failure" assumption). The above reasoning steps to exonerate an abnormally-behaved module are called the **Causal-Backtracing principle.** With C exonerated, we infer that since B is enabled by A and A is normal, B's abnormality must be due to its own failure (by the "uni-directional" assumption). The failure of the PL mechanism is thus postulated to be functional module B. We call such reasoning steps the **Failure-Identification principle.** At this point, applications of the two diagnosis principles illustrated are local in nature, namely, the related information they utilize always comes from direct neighbors of a module. We now discuss how non-local information can be incorporated in their deduction processes.

Because of the limited availabilities of sensors, we may not be able to determine the operational status of each module. By the single-failure assumption, we can extend the causal-backtracing principle to achieve indirect failure associations. For example, in a PL mechanism as shown in figure 3.2(b), by applying the extended version of the backtracing principle, C and D are exonerated even if the status of C is unknown. Again, by

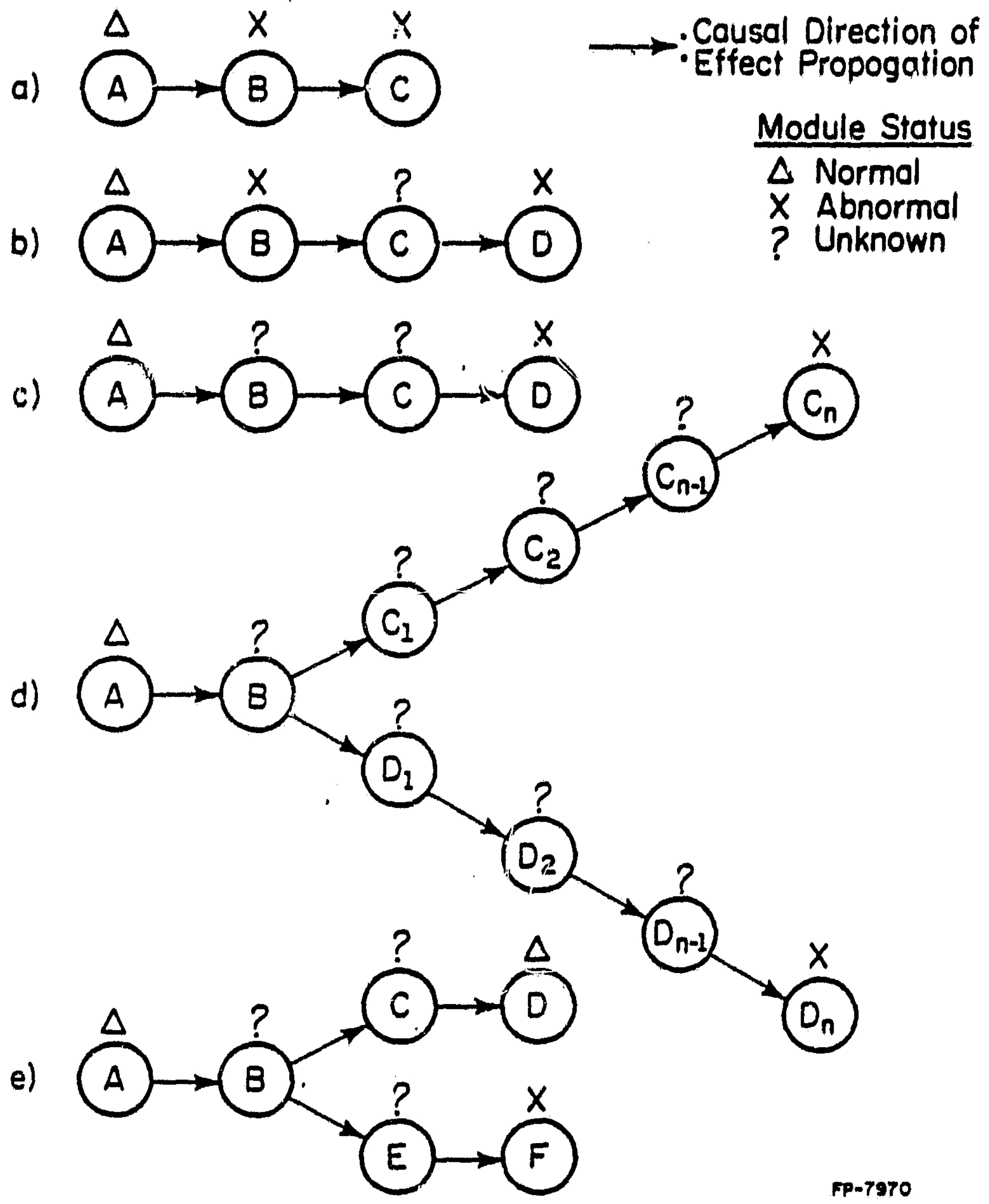


Figure 3.2 Examples to Illustrate Fault-Isolation Principles

the fault-identification principle, B is postulated to be the culprit. However, not in all cases can the isolation process localize one specific faulty module. In a PL mechanism with a similar causal structure but without the operational status of B, as shown in figure 3.2(c), a set of possible faulty modules, namely, B, C, and D, is postulated instead.

We now explore a more sophisticated diagnosis principle, called the **Common-Cause principle,** which takes the advantage of special causal-fanout structures existing in the PL-model of a mechanism. In a situation as shown in figure 3.2(d), the fact that abnormal Cn and abnormal Dm share the same causal "upstream" module B imply that the failure is due to a common cause upstream, namely, B. In this case, by further applying the fault-identification principle on A and B, we postulate that B is the culprit even with no knowledge about the operational statuses of C1...Cn-1 and D1...Dm-1.

The diagnosis principles developed so far emphasize the direct and indirect uses of module-abnormalities (symptoms) in postulating possible failures. We now discuss the **Implied-Exoneration principle** which excludes some functional modules from failure possibilities by reasoning with normal (or no-fault) measurements. To be able to apply the implied-exoneration principle, we introduce an "intentional design" assumption which states: "a functional module will not operate normally unless all its enabling-preconditions are satisfied". This means, if a module fails, all modules causally-enabled by it will show symptoms if sensor-measurements are available. Also, no module should have implicit redundancies built in[3]. Potential applications of the implied-exoneration principle are illustrated by an example, as shown in figure 3.3. In the PL-modeled mechanism with causal relationships as shown in figure 3.3, knowing that functional module F is abnormal prompts the postulation of a failure set which includes A, B, C, D, G, and H by applying the fault-identification principle. However, knowing F is operating normally causes us to assert, by the implied-exoneration principle, that A, B, and C must be operating normally even though no direct measurements are available. (D may be partially faulty, namely, it may have a normal output to E while fails in the output to G.) Therefore, by applying the implied-exoneration principle, the fault-isolation process postulates a smaller possibility-set with D, G, and H being potential culprits.

In reviewing Brown's WATSON approach for the troubleshooting of radio-receivers [3][4], we find it is a simple case of our fault-identification process, namely, it is a straightforward application of the identification principle alone to a near-ideal mechanism: a radio-receiver with stage-decomposed modules.

---

[3] In case of a real-world mechanism in which redundancies do exist, they should be made explicit in the causal description. The detail will be discussed within the context of flow-causality models.

[4] The author must point out that WATSON's reasoning is for the generations of test-points, which implicitly assumes that the status of each functional module can be determined. Thus, our comment is not about the weakness of his approach, rather, is about the generality of his theory.

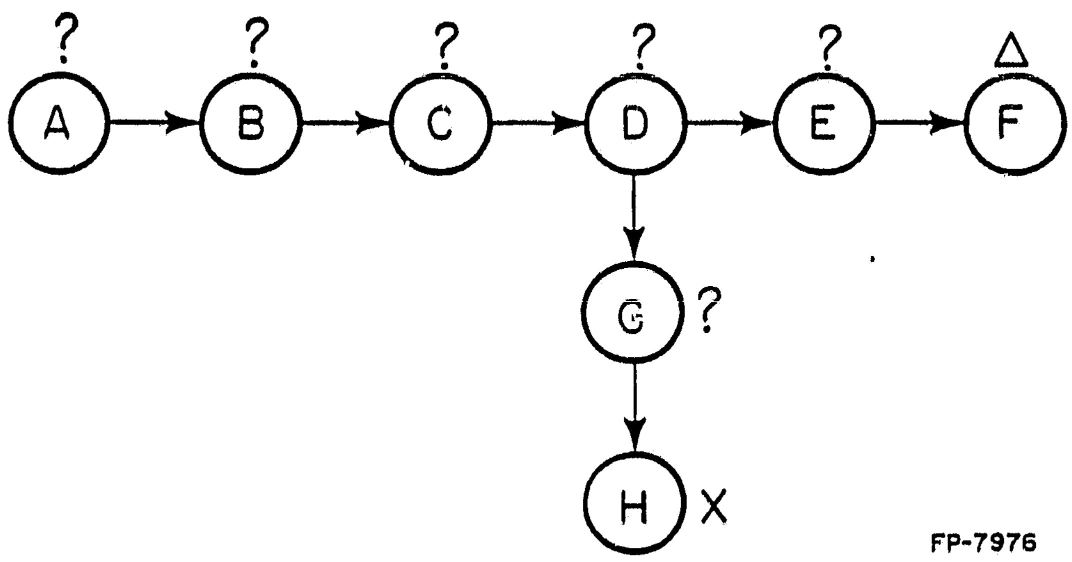


Figure 3.3 Examples to Illustrate Fault-Isolation Principle (cont.)

3.2.2. Fault Isolation in a Real-World Analog Mechanism

The study of failure isolation on the ideal mechanism serves as the foundation for the understanding of failure isolation in a non-ideal real-world mechanism. Summarizing the results from previous discussions, the four diagnostic principles are developed based on following assumptions made on an ideal PL mechanism:

[a] Single-Failure Assumption -- Only one failure can occur in a mechanism at a time.

[b] Uni-Directional Assumption -- Propagation of causal effects among functional modules are "uni-directional". Only one failure occurs at one time.

[c] Status-Determinable Assumption -- There exists a mapping by which the normal/abnormal operational status of each functional module can be determined if some sensory measurements are available.

Applying these four diagnosis principles to non-ideal mechanisms results in a heuristic (rather than exclusive) set of failure hypotheses in the sense that the actual failure will be included in the possibility set most of the time -- exceptions are due to violations of ideal assumptions on the PL mechanism. To layout a practical approach for failure isolations in a non-ideal real-world mechanism and to understand the limitation of such approach, it is most helpful to discuss our fault-isolation methodology by studying the conditions under which these ideal assumptions are violated in an analog mechanism.

The single-failure assumption is consistent with the purpose of our hypothesization task, which is to postulate a set of primary-failure possibilities. With respect to the second assumption of uni-directional casual propagation, we rarely find any real-world mechanism which can satisfy this assumption without exceptions. Therefore, we propose a flow-causality model which organizes the functionality of an engineered mechanism into causally-related flow-groups. The intention is to provide a functional decomposition of the overall mechanism into modules of flow-subsystems with near-unidirectional inter-module interactions. Within a flow-group, variables are closely-associated in the sense that any failure "almost" certainly will affect all variables in the same flow subsystem. Such property of close associations among variables also makes the condition favorable to apply the four diagnosis heuristics since the operational status of each functional module is likely to reflect on any given sensor within the flow subsystem. Interesting as the idea may sound, we have to stress at this point that there exists counter-examples in real-world mechanisms which will conditionally violate the basic assumptions of the PL model. Therefore, the effectiveness of applying the four diagnostic heuristics to a real-world mechanism depends on how often the ideal assumptions are violated -- which, in author's opinion, is an engineering-oriented issue. For this reason, the author emphasizes the engineering aspect of our hypothesization methodology in terms of its usefulness, rather than its theoretical importance.

3.2.2.1. Functional Decompositions and the Flow-Causality Model

Our purpose of developing the flow-causality model of a mechanism is to create a functional description which is organizationally similar to that of the Production-Line model in the hope that the isolation principles developed for the ideal mechanism can be similarly applied for the generation of fault hypotheses.

Intuitively, the functionality of an engineered mechanism can be described as a set of causally-related subsystems which implements the intention of its designer. For example, the mechanism of a simple jet-airplane can be functionally organized into following subsystems: the fuel subsystem, the engine subsystem, the oil subsystem, the electrical subsystem, and the hydraulic subsystem. Following the intention of airplane engineers, the causalities among the subsystems are, as shown in figure 3.4, such that the fuel subsystem drives the engine subsystem, and the engine subsystem in turns drives the oil subsystem, the electrical subsystem, and the hydraulic subsystems.

In order to activate each subsystem, the designer arranges a set of components in such a way that some "products", or called "medium" can be forced to other causally-depended subsystems. With this view, we can abstractly characterize a functional subsystem as a "flow subsystem" which creates and transfers a particular type of medium to other subsystems, and the sensors in a subsystem serve to monitor the "potential-of-flow" or the "rate-of-flow" at various points of a flow

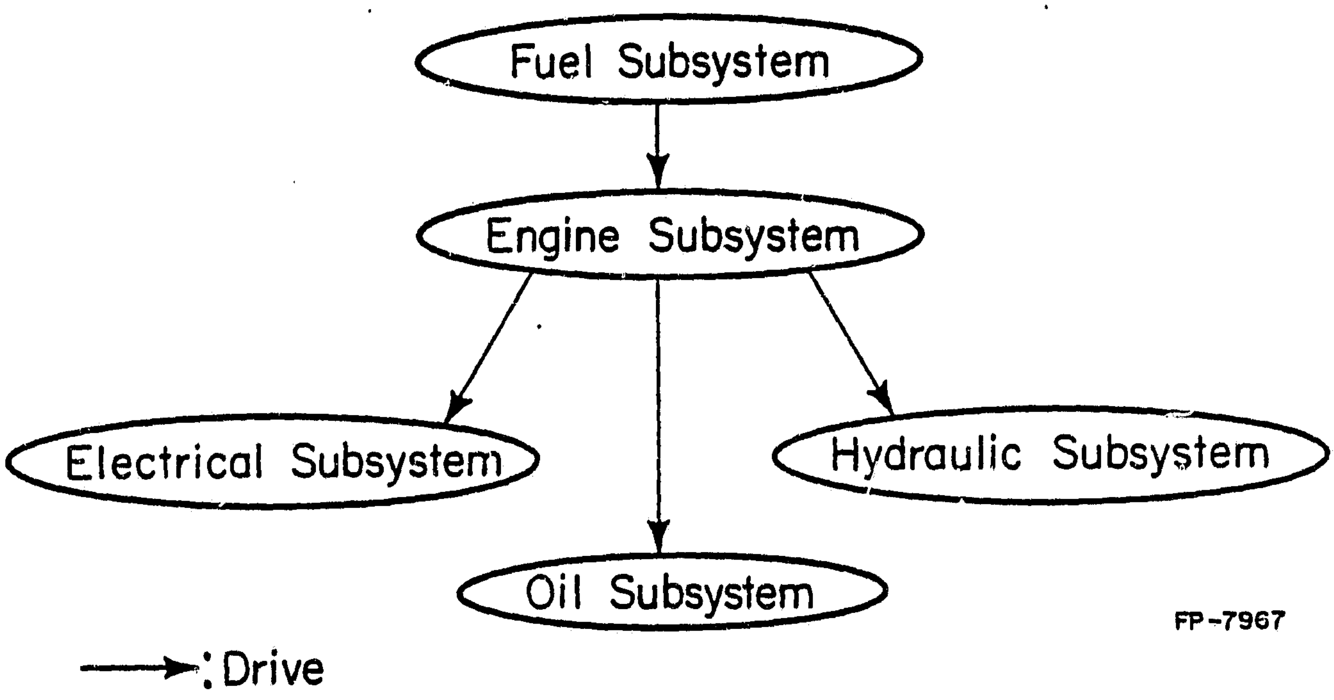


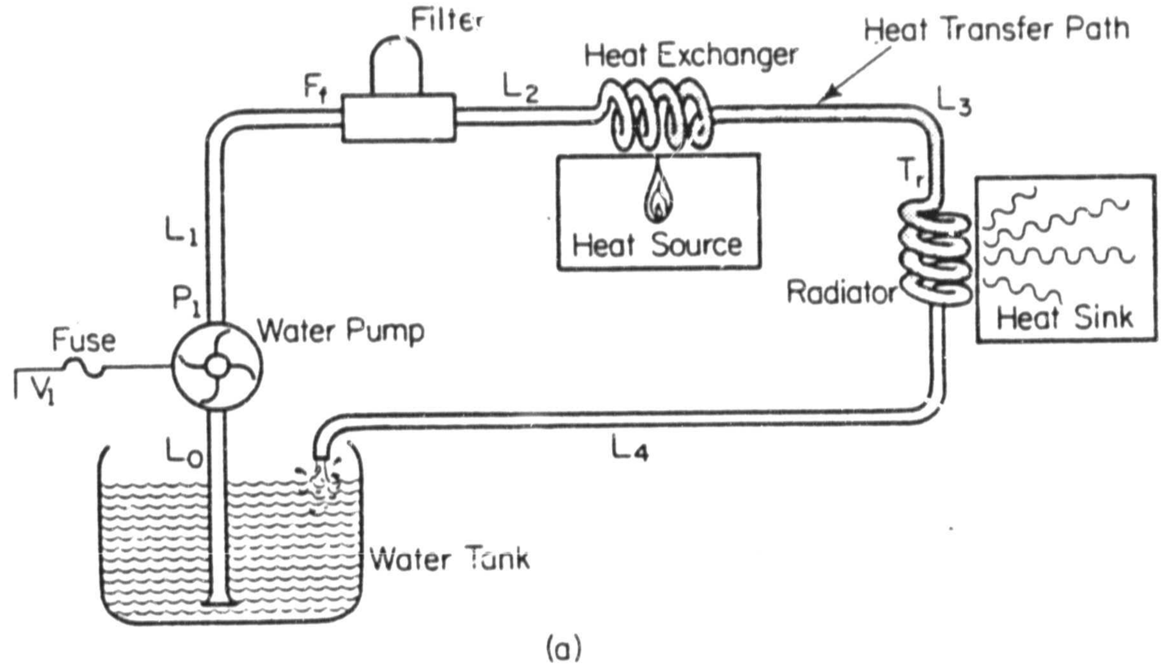
Figure 3.4 Causal Dependencies among Subsystems of an Airplane Mechanism

subsystem. In some types of flow subsystems, the medium is stored (as in an oil subsystem) rather than created (as in a electrical subsystem), thus the "quantity-of-medium" (as the oil-quantity) becomes relevant. Some typical flow subsystems and their corresponding media are listed below:

Types of Flow-Subsystems	Flow Media
Oil Subsystem	engine-oil
Hydraulic Subsystem	hydraulic-fluid
Electrical Power Subsystem	electricity
Thermal-Transfer Subsystem	heat
Pneumatic Subsystem	air

An interesting observation is that the flow-variables within a flow subsystem are closely-associated in the sense that there **seldom** is a case that symptoms of a failure show on some, but not all, of the flow variables. In contrast, variables belong to two causally-related subsystems are likely to be unidirectional-associated in the sense that a failure in the "driven" subsystem **usually** will not reflect on the "driving" subsystem.

Finally, we discuss representational issues of the flow-causality model by an example of cooling mechanism, commonly used in the central air-conditioning system, as shown in figure 3.5(a). As shown in figure 3.5(b), the cooling mechanism is functionally decomposed into three flow-subsystems: the electrical-subsystem, the water-circulation subsystem, and the heat-removing subsystem, with electricity, water, and heat as flow-media respectively.



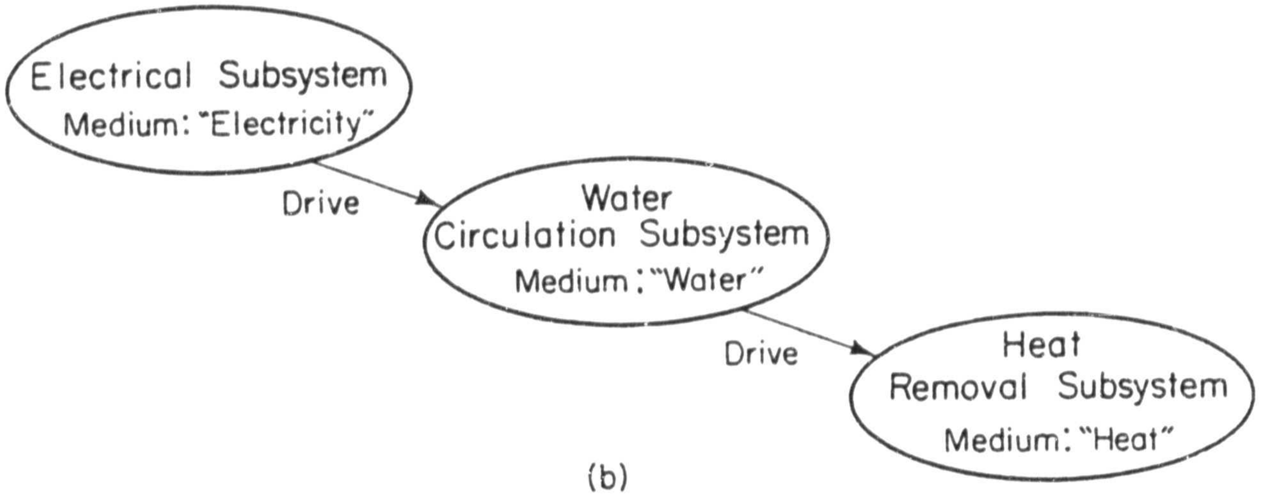


Figure 3.5 A Cooling-Mechanism Example for the Flow-Causality Model

The causalities among these three subsystems are described as following:

```
(drive 'ELECTRICAL-SUBSYSTEM 'WATER-CIRCULATION-SUBSYSTEM)
(drive 'WATER-CIRCULATION-SUBSYSTEM 'HEAT-REMOVING-SUBSYSTEM)
```

For each flow-subsystem, we further establish the association between each function roles and physical components by which the function is actually implemented. As shown in figure 3.6, the abstract functional role "reservoir" is implemented as the "WATER-TANK", the "booster" as "WATER-PUMP", etc. The "delivery-path" is implemented by a group of components which structurally are in series. A frame-like representation is adopted to describe the subsystem model:

```
subsystem-name: 'WATER-CIRCULATION

abstract-functional-template: 'HYDRO-FLOW-CIRCULATION

medium: 'WATER  /* a-kind-of non-compressible fluid */

functional-roles:  /* mapping to the structural components */

    reservoir: 'WATER-TANK
    supply-path: 'L0
    booster: 'WATER-PUMP
    delivery-path:
      (in-series L1 FILTER L2 HEAT-EXCHANGER L3 RADIATOR L4)
    drain: 'WATER-TANK
```

Notice that the specification in the "abstract-functional-template" slot serves as a linkage to a body of knowledge about

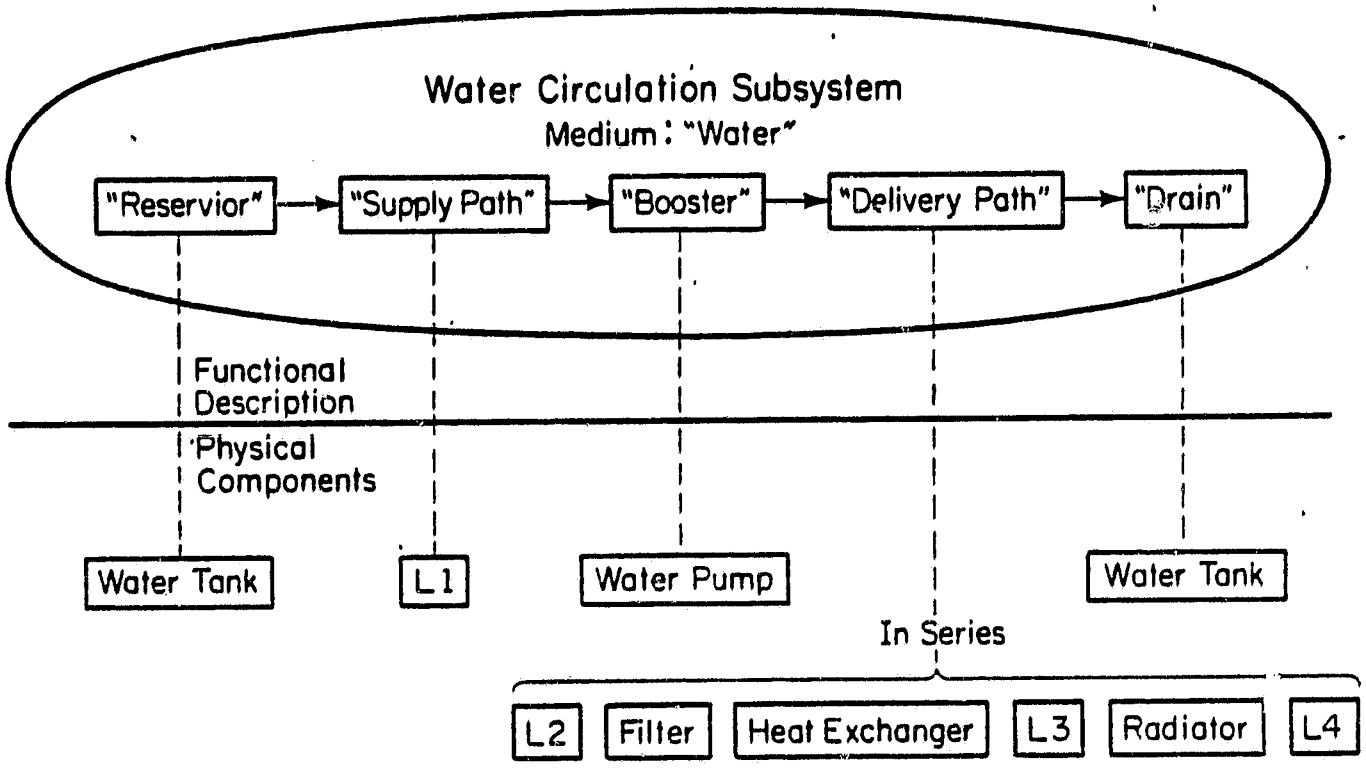


Figure 3.6 Functional-Structural Association in a Water-Circulation Subsystem

a general non-compressible hydro-flow circulation subsystem which this particular WATER-CIRCULATION-SUBSYSTEM is an instance of. To be included in this knowledge chunk is a piece of knowledge to map observed symptoms into a set of failure hypotheses[5].

In a mechanism with built-in redundancies at the subsystem level, these special relationships will be explicitly represented in their causal descriptions. For example, in a modern DC-10-like [4] airplane, the fuel subsystem, the electrical subsystem, and the hydraulic subsystem are all designed with heavy redundancies, as shown in figure 3.7. As a typical case, the casual relationships between the power-plants (engines) and the electrical subsystem will be described as:

```
(drive
   (or
      'POWER-PLANT-1
      'POWER-PLANT-2
      'POWER-PLANT-3
   )
   'ELECTRICAL-SUBSYSTEM
)
```

[5] One simple approach is to attach a pre-coded procedural knowledge to the frame of abstract flow-subsystem so that it can be inherited by all flow-subsystems of the same type. Our emphasis is on the organization of the flow-subsystem knowledge. The failure-postulation procedure, because of its engineering nature, is not discussed in this thesis. An illustration of its applications will be discussed later in an example.

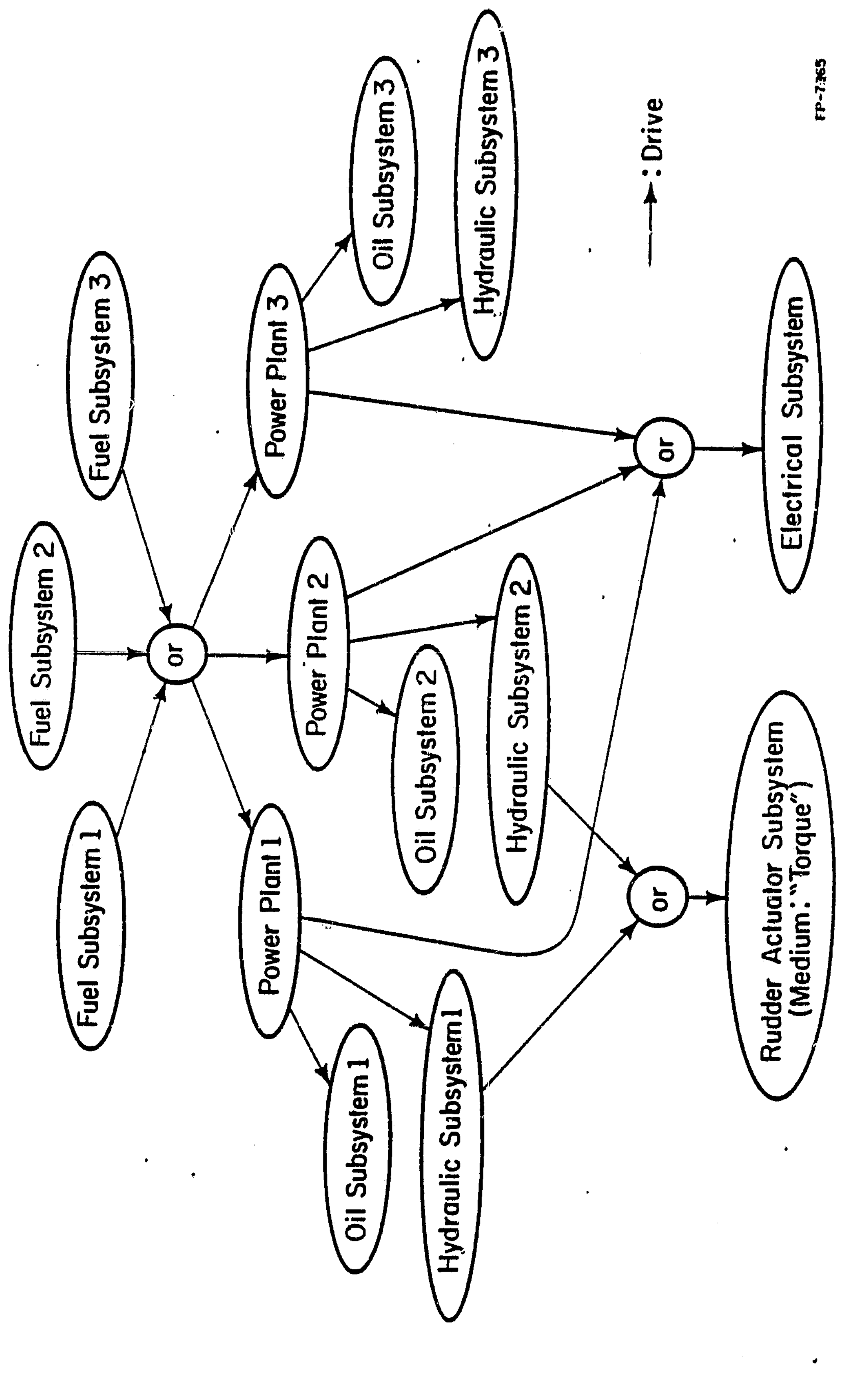


Figure 3.7 An Explicit Causal Representation for a Redundant Mechanism

3.2.2.2. Failure Hypothesization with the Flow-Causality Model

The task of failure hypothesization is achieved in two steps: **subsystem isolation** and **failure postulation.** In this section, we illustrate the process of failure hypothesization by the example of water-cooling mechanism, shown in figure 3.5.

First, we deal with the problem of isolating the faulty subsystem. The subsystem-level of a flow-causality model provides a description of causal relationships among subsystems analogous to that of the ideal PL-modeled mechanism. As a precondition to applying the four isolation principles developed for the ideal mechanism, we need to define a predicate by which the operational status of each subsystem can be determined from its related sensors, which happen to be available. Knowing that sensors are installed in an operational mechanism to monitor flow-related variables, thus, by the flow-causality model, available sensors are divided into groups according to associations of their corresponding variables. The operational status of each subsystem module is defined as follows: (1) the status is said to be normal if all available sensors in a flow subsystem have normal readings, (2) the status is said to be abnormal if any sensor in the subsystem shows deviation from its normal value, and (3) the status is unknown if no sensor-reading is available. With the definition of the subsystem status, the four isolation heuristics can be applied to isolate the faulty subsystem. In the example of a cooling mechanism, given that P2 and T3 readings are deviated from their normal range but V1 is normal, we heuristically isolate the failure to the water-circulation subsystem by applying the causal-backtracing and the fault-identification principles.

Second, with the focus shifted to a particular flow-subsystem, a set of failure hypotheses is postulated with the application of the procedural knowledge inherited from the abstract flow-subsystem to which this flow-subsystem is an instance of. The failure-postulating procedure only concerns itself about interpreting variables which are related to this flow-subsystem. Continuing the cooling mechanism example in figure 3.5, from the sensor-observation that P1 is normal, it exonerates three functional roles: the booster, the supply-path, and the reservoir, from being considered as possible culprits. Thus, the failure, if it indeed occurs within this flow-subsystem, must come from the delivery-path or the drain (which is excluded in this case since it is the open space in the WATER-TANK). Interpreting a low flow-rate (Ff) detected in a serial path (L1 -> FILTER -> L2 -> HEAT-EXCHANGER -> L3 -> RADIATOR -> L4), following two rules can be applied:

(1) Upstream-leakage -- any component upstream to the point of detection can be leaking to cause the low flowrate measurement.

(2) Path-cloggage -- any component in the flow path can be clogged to cause the detected low flowrate.

As the result of applying these flow-tracing rules, the following possible failures are postulated: L1 leakage, L1 cloggage, FILTER cloggage, L2 cloggage, HEAT-EXCHANGER cloggage, L3 cloggage, RADIATOR cloggage, L4 cloggage. To re-emphasize the significance of our hypothesization-and-verification approach, we stress that the "L1 leakage" is an outstanding case during the verification phase due to its unique time-elapsed effects. Among the above failure hypotheses, the predictive analysis process will conclude that only the "L1 leakage" assertion will be expecting further symptom-event: the dropping of P1 to zero after some time delay because the water-tank will eventually become empty. If this symptom indeed is observed over the P1 sensor some time later, the diagnosis process can conclude that the "L1 leakage" is the culprit.

3.3. Conclusion

As the conclusion of our study, we propose a future computer-based intelligent diagnosis system be organized as shown in figure 3.8. The time-continuous sensory data is first abstracted into a sequence of qualitative symptom events, which will be compared with the predicted failure behavior from the intelligent diagnosis program. Those accepted failure hypotheses, together with their justifications, will be (1) presented through some means of man-machine interfaces (e.g., graphical or text display, voice synthesizer, etc.) in a form which can be readily understood by people so as to help them in their decision-making[6], (2) presented through a shared

[6] We call this feature the intelligent interface since it presents the diagnostic result with concise justification, rather than just presenting the raw data.

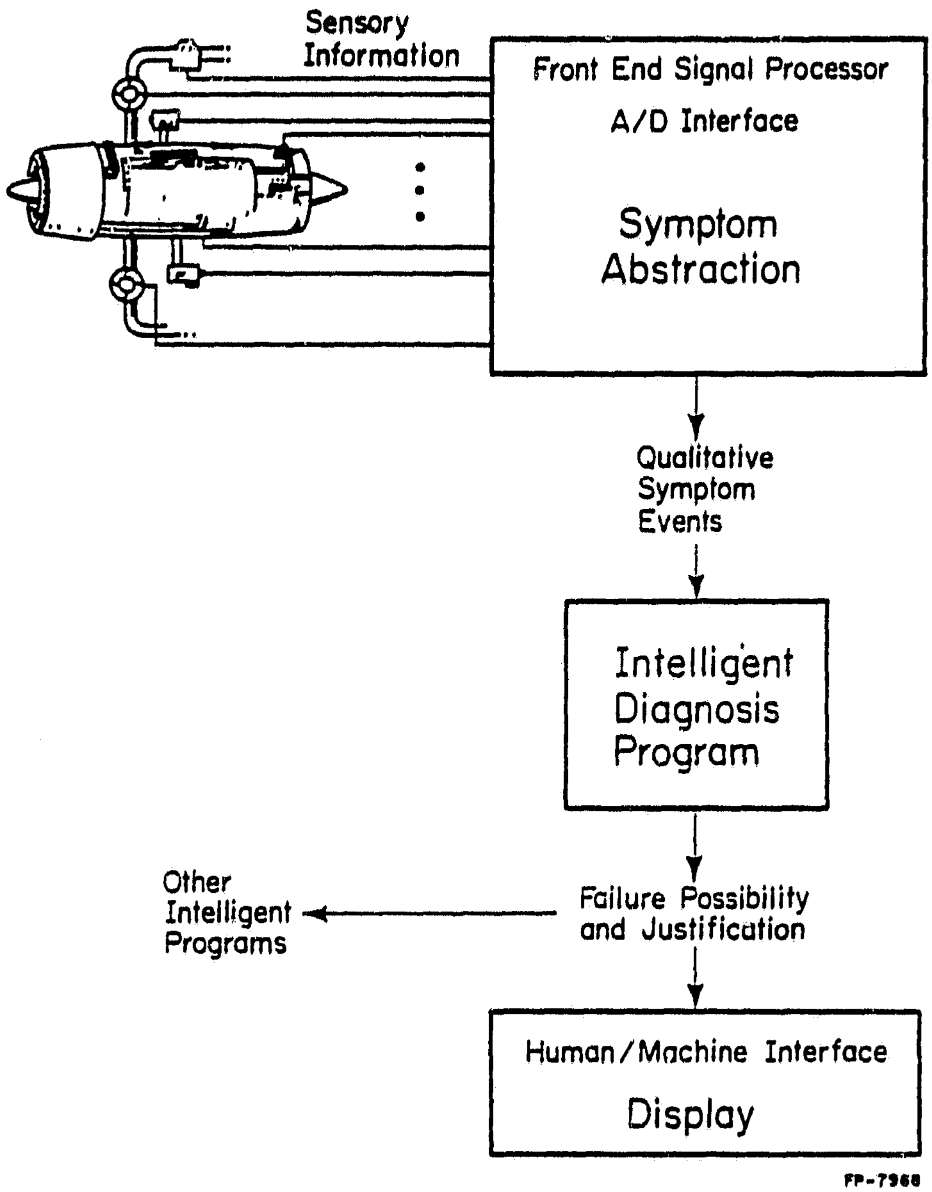

Sensory Information
Front End Signal Processor
A/D Interface
Symptom Abstraction
Qualitative Symptom Events
Intelligent Diagnosis Program
Failure Possibility and Justification
Other Intelligent Programs
Human/Machine Interface
Display
FP-7960

Figure 3.8 System Organization of a Future Intelligent Diagnosis System

data base in its entirety to other intelligent programs so as to allow other intelligent activities (e.g., recovery-planning) to proceed.

REFERENCES

[1] Pan, Y. (1983). Qualitative Reasonings with Deep-Level Mechanism Models for Diagnoses of Dependent Failures. Ph.D. Dissertation. University of Illinois, Urbana, IL. CSL Report T-132.

[2] van Melle, W. (1979). A Domain-Indenpendent Production-Rule System for Consultation Programs. Proceedings, 6th International Joint Conference on Artificial Intelligence, Vol. 2.

[3] Brown, A. (1977). Qualitative Knowledge, Causal Reasoning, and the Localization of Failures. Ph.D. Dissertation. M.I.T. AI-TR-362.

[4] (1975). DC-10 Flight Crew Operating Manual. McDonnell Douglas Corporation.