REPORT T-117 AUGUST, 1982

COORDINATED SCIENCE LABORATORY

MULTILEVEL SEMANTIC ANALYSIS AND PROBLEM-SOLVING IN THE FLIGHT DOMAIN

FINAL REPORT

NASA GRANT NCCI-52
JULY 11, 1981 - JULY 10, 1982

BY:
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 NCCI-52

July 11, 1981 - July 10, 1982

Submitted by

Professor R. T. Chien
Principal Investigator

Prepared by

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

Table of Contents

Chapter I

Introduction and Summary

In this report we summarize the progress resulted from the NASA Cooperative Agreement NCCI-52 on the subject matter of "Multilevel Semantic Analysis and Problem-Solving in the Flight Domain". This work covers the period from July 11, 1981 to July 10, 1982.

The overall goal of this project is the conceptual development of a computer-based cockpit system which is capable of assisting the pilot in such important tasks as monitoring, diagnosis and trend analysis. The system is properly organized and is endowed with a knowledge base so that it enhances the pilot's control over the aircraft while simultaneously reduces his work-load.

The first phase of our work deals directly with the monitoring function. Based on a novel hierarchical levels model the monitoring function is achieved via the generation of a dynamic reference which is context-based. The planning algorithm produced a desirable plan at each level and details of the plans are generated as the propagation of the planning activity progressed top-down from the route level, passing through the trajectory level to reach the control level. Plan recovery activities will be needed whenever a change occurs in the context. Permissible changes include weather, controller commands or system malfunctioning. Details of this work is summarized in Chapter II of this report.

A second phase of our work is in the automatic diagnosis of system malfunctioning based on sensory data. Since system redundancy normally provides protection against single fault our work emphasizes the real-world problem of diagnosis of multiple-fault situations with fault masking. With the use of flow model analysis the fault is isolated to certain subareas where functional models are then used to deduce consistancy of assumed fault patterns. This phase of our work is discussed in detail in Chapter III of this report.

The final phase of our work deals with the rationalization of structures. This is needed for the reasoning of mechanisms for the purposes of diagnosis. One of the major weaknesses of present theory of diagnosis is its shallowness in understanding the functions of the mechanism its intends to diagnose. A theoretical understanding of how mechanism work is a fundamental precondition for intelligent deep-level diagnosis.

Chapter II

The Intelligent Monitor

1. Introduction

In this past year much progress has been made on the intelligent flight monitor research. The most important progress was made in the development of the conceptual levels planning architecture. This architecture is the culmination of the work on the multi-level planning theory [1]. The conceptual levels planning architecture is necessary because intelligent monitoring requires sophisticated planning capability.

The function of the computer monitor is to continuously observe the flight environment and evaluate the situation for possible errors that would threaten safety of the flight. The use of such an onboard computer monitor can significantly reduce the workload of the flight crew by relieving the crew of the tedious and repetitive task of scanning the numerous instrument readings for possible problems. The computer monitor would be especially useful during periods of high workload when the crew is busy. By assisting the crew in the monitoring task, the computer monitor enables the crew to devote more time to other time-critical tasks. Another advantage of the computer monitor is that the monitor would not be affected by typical human failings such as boredom, fatigue, or fixation. It is for these reasons that commercial flight crews recommended the monitoring task for the intelligent onboard computer.

Monitoring the activities of the flight crew requires knowing what the flight crew should be doing at each point of the flight. In other words, the monitor requires a reference of how the world should be in order to determine if the world is as it should be. Generating this reference is a planning task. It is necessary to endow the monitor with the knowledge of planning and executing the flight. Automatic planning in the flight domain is a formidable task. Firstly, the flight domain is a complex domain. Flight requires the knowledge of route planning, navigation, aircraft control, emergency procedure, and aircraft subsystems. The coordination of these different knowledge is complicated by the fact that they often interact with each other [2,3]. Secondly, the flight domain is also a dynamic domain. Events beyond the control of the flight crew affect the flight. Weather condition may change quickly and mechanical equipments both on the ground and on the air may fail. Thus the carefully devised plan may be ruined by dynamic events. Any planner operating in the flight domain must deal with the complexity of this domain. The conceptual levels planning architecture is designed toward this goal.

The conceptual levels architecture organizes the domain knowledge into conceptual levels. A conceptual level contains a subset of the domain knowledge and is related to other levels by the form/function and the precondition inter-level relationships. The levels form a hierarchy based on these inter-level relationships. Planning in the conceptual levels architecture consists of activities within a level and activities between the levels. Inter-level planning controls the intra-level planning at each level and together with the levels hierarchy provides the global viewpoint necessary to control the domain knowledge complexity.

2. Motivation for the Conceptual Levels Theory

The intelligent monitor requires dynamic references for the many variables of the flight domain. These references are generated by the planner. Planning in the flight domain is a formidable task. Though much work has been done in automatic planning, none of these works employ a domain as complicated as the flight domain. The planner in the flight domain must deal with the complexity in the horizontal direction as well as the complexity in the vertical direction.

Horizontal complexity is the sheer number of variables that must be considered. These variables range from the aerodynamic variables such as the angle of attack, the pitch angle, the climb rate, the velocity, to the subsystems variables such as the engine rpm, the fuel flow rate, the engine temperature, the bus switch setting, the fuel valve setting to the aerodynamic control variables such as the elevator setting, the aeileron setting, the landing gear control setting, the flaps setting to the navigational variables such as the aircraft location, the aircraft altitude, the aircraft heading, the VOR frequency setting, and the refueling airport. The sheer number of variables makes it difficult for the planner to determine which variable should be considered next.

The flight domain is also complex in the deep (vertical) sense. The flight domain has many facets that interact in an intricate fashion. The aircraft climb rate is dependent on the flap setting, the elevator setting, and the throttle setting. The throttle setting is implicitly dependent on the engine system. The engine system, in return, is dependent on the pitch angle and the elevator setting since the engine temperature is dependent on the pitch angle and the throttle setting. The variables have a tangled relationship with each other. These tangled relationships between the variables make it difficult for the planner to determine what is important at a given point of planning.

Besides the domain complexity, the planner is must also deal with a dynamic domain. All the planning works thus far have dealt with static domains where the planner is the only agent that can change the world. This is not true in the flight domain. The flight domain is inherently dynamic. The weather may deviate from the forcast unexpectedly. The crew may be slow in correcting errors or may actually deviate from the flight plan. Lastly, the aircraft itself may fail in some way, thus degrading the aircraft's capability. The dynamic flight domain greatly complicates the planner's task since a carefully planned plan may fail due to factors outside the planner's control. Thus the planner must be able to initiate planning with incomplete information and be able to correct plan failures caused by external events.

Any planner operating in the flight domain must deal with the complexity of this domain. It is imperative that this complexity be controlled. The conceptual levels planning architecture is designed toward this goal. The conceptual levels architecture is a refinement of the previous multi-level architecture. It is also the descendent of the hierarchical planner [4,5,6,7,8].

3. The Conceptual Levels Theory

The conceptual levels approach is a semantic approach to obtaining higher-level planning direction. The conceptual levels approach is descended from the hierarchical planning approach. The hierarchical planner plans abstractly using a simplified model of the domain. It then gradually fills in the less important details. The conceptual levels planner augments this definition in that the hierarchy is not based on the amount of the details but rather the kinds of details. Instead of the less details of the abstraction space, the conceptual levels contains different kinds of knowledge. In the conceptual levels hierarchy, the semantics change as well as the amount of detail.

The conceptual levels approach organizes the domain knowledge into levels. Planning is done within a level and between levels. The levels partition the domain knowledge into smaller partitions, but the partitions (levels) also relate to each other teleologically. The levels also form a levels hierarchy. The relationships between two levels can be either the form/function relationship or the precondition relationship. These inter-level relationships form the basis for higher-level viewpoint.

3.1. The Causal Framework

The planner operates in a world of causal relations. The variables in the world are related to each other through these causal relations. The planner examines these causal relations to generate actions that will maneuver the goal variables to the desired state. The conceptual levels architecture is motivated by the causal framework observation. The causal framework observation is that the variables of a domain do not relate to each other with the same intensity. In other words, some variables are more closely related than others; some variables are tightly related while others are loosely related. A causal framework is a group of tightly related variables and the causal relationships between these variables. Figure 1 illustrates the causal framework organization of the domain variables.

A conceptual level is associated with a causality framework. By decomposing the domain into causality frameworks, the domain is simplified into nearly independent subdomains. A conceptual level is more than a group of variables and causal relations. A conceptual level is a planner, a representation, and the knowledge to communicate with other conceptual levels. The conceptual levels form a hierarchy that defines the first cut of problem decomposition and defines the relationships between the subproblems. The conceptual levels hierarchy defines the nearly independent subproblems and how they interact with each other at the interface. Besides functioning as a means of controlling complexity, the hierarchy also structures the knowledge base. Each level has its own knowedge base of flight knowledge and the knowledge of interacting with other levels.

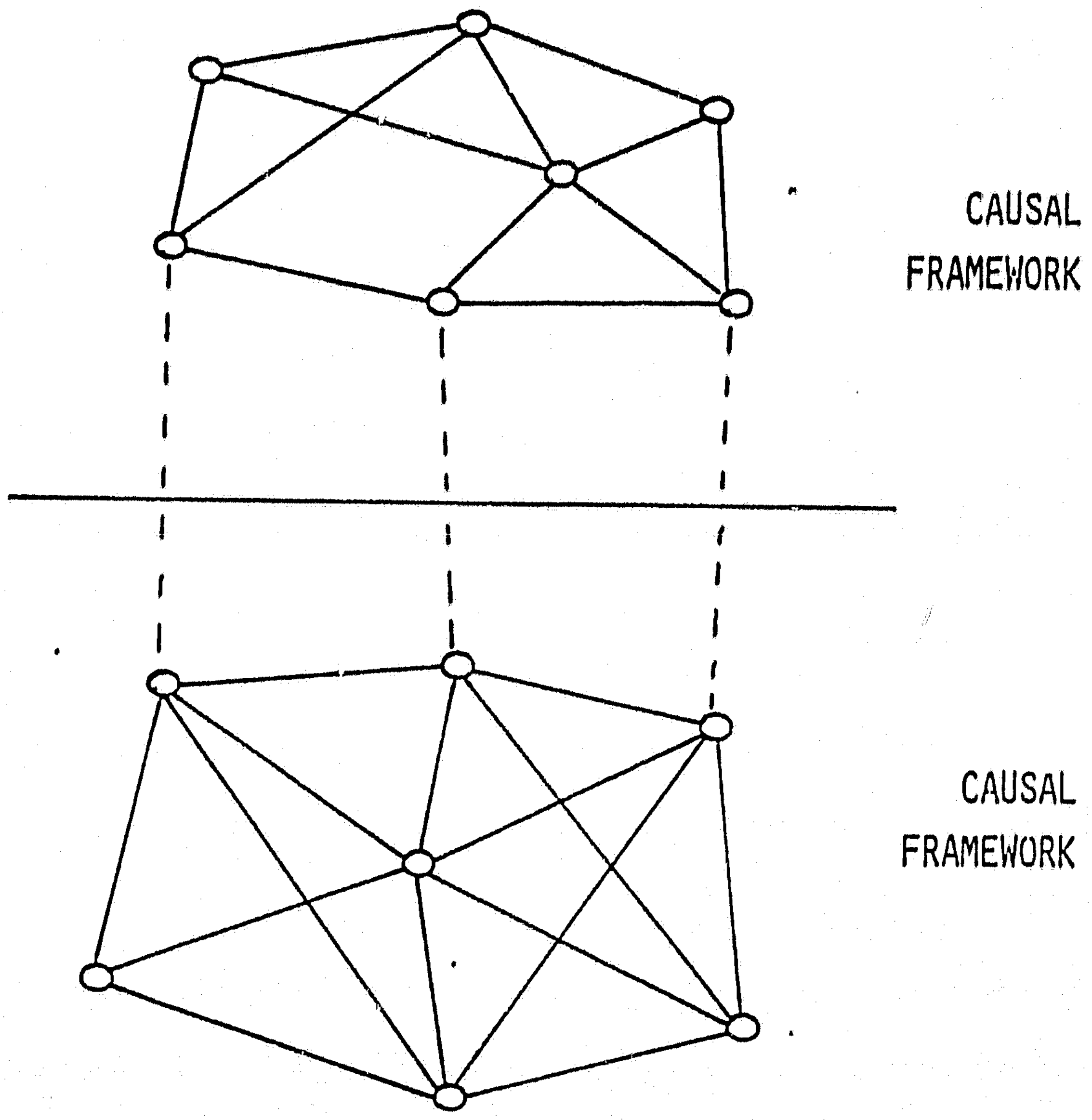


Figure 1 The causal framework organization of domain

3.2. The Levels

The flight domain knowledge is presently organized into four conceptual levels: the route level, the trajectory level, the flight-control level, and the aircraft subsystems level. Figure 2 illustrates the hierarchy. The route level, the trajectory level, and the flght-control level form a form/function hierarchy with the route level at the top and the flight-control level at the bottom. The form/function relationship between two levels is such that a complete plan at the top level (the form level) can be implemented at the bottom level (the function level) with the variables from the bottom level. An example of the form/function relationship is that a computer register is implemented by flip-flops which are implemented by logical gates which are implemented by electronic circuits. Another example is that an aircraft route is implemented by a trajectory which is in turn implemented by a sequence of flight control settings. The second kind of inter-level relationship is the precondition relationship. Here, instead of implementing the upper level, the lower level enables or supports the upper level. An example of this relationship is that the power supply enables the electrical circuits to function and indirectly enables the registers to function. Another example is that the engine system enables the throttle to be effective. The electrical system also enables navigation which makes the flight controls settings sensible. The subsystems level supports the flight-control level. The subsystems level also supports the trajectory level (for navigation).

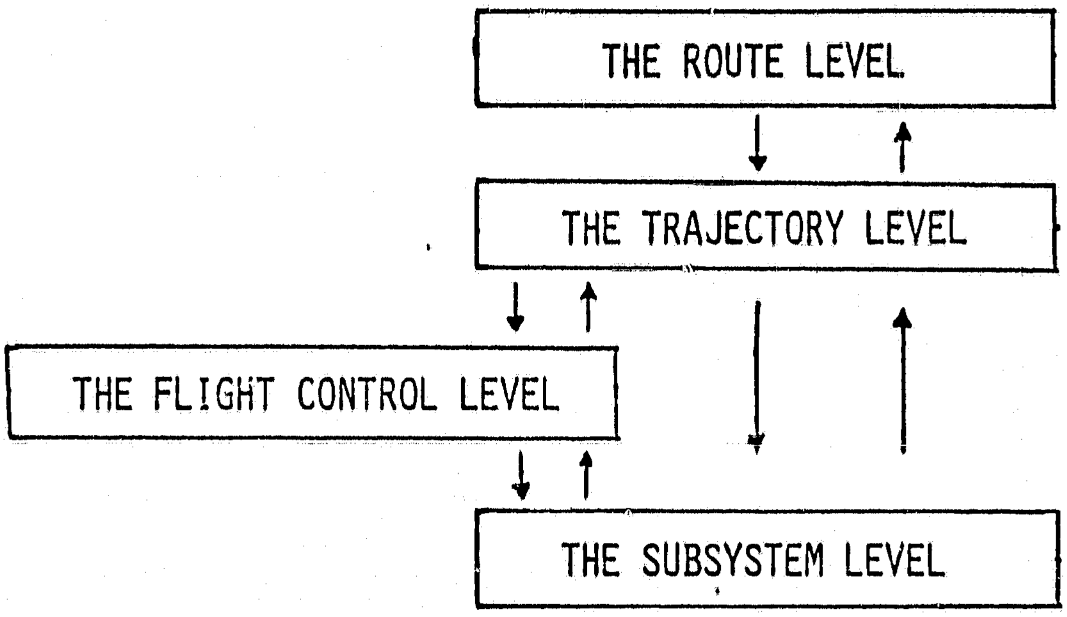


Figure 2 The conceptual levels hierarchy

3.2.1. The Route Level

The route level is the highest level in the conceptual level hierarchy (combined form/function and precondition hierarchy). The route level is the highest level because it is the most abstract level and because it has the broadest viewpoint over the plan. The route level is responsible for planning a route from the origin airport to the destination airport. The route is a sequence of airway segments. An airway segment is a segment between two navaids, typically a vortac or a non-directional beacon. Since it is common to have a navaid near an airport, the route segment can also terminate at an airport.

At the route level the world is abstracted to a network of nodes and links. The nodes represent the airports and navaids and the links represent the airway segment between the two nodes. Other information are associated with these nodes and links. Examples are the aviailability of the airport and the airway segments, the refueling capability of the airport and the runway length, the adverse weather position and velocity, and the minimum enroute altitude of the airway segments. The knowledge base also contains knowledge of the aircraft such as the aircraft airspeed, the aircraft service ceiling, and the aircraft range.

The route level also contains the active knowledge necessary to generate the route. Planning at the route level is essentially a constraint satisfaction problem. A plan is a sequence of airway segments that leads to the destination. Besides achieving the goal, the route must satisfy a host of constraints. These constraints can be stated as the preservation of the aircraft integrity, adherence to the FAA regulation, and the minimal expenditure of fuel and time. These basic constraints can be decomposed to other constraints. For example, minimal fuel expenditure can be expanded into short route, low power setting, best altitude, and no loitering constraints. Given this formulation the route-level planning is based on a constraint-guided search. The search is first guided by the more inflexible constraints to obtain plausable planning islands. Then more flexible constraints are applied to connect these planning islands.

The route level generates a route consisting of a squence of airway segments. Figure 3 gives an illustration of the planning at the route level. This route is passed to the lower levels. Besides the route, there is another bidirectional interface with the lower levels consisting of the aircraft performance variables such as the airspeed, service ceiling, and the range. If values of these interface variables are unacceptable to the lower levels, replanning at the route level will be necessary.

3.2.2. The Trajectory Level

The trajectory level is the conceptual level below the route level. The trajectory level generates a 3-dimensional flight trajectory that extends to the destination. In order to plan its plan, the trajectory-level planner needs direction from the route level. A completed route-level plan is passed to the trajectory level with the proper semantic transformation. A semantic transformation is sometimes necessary for communication between levels because the levels may use different

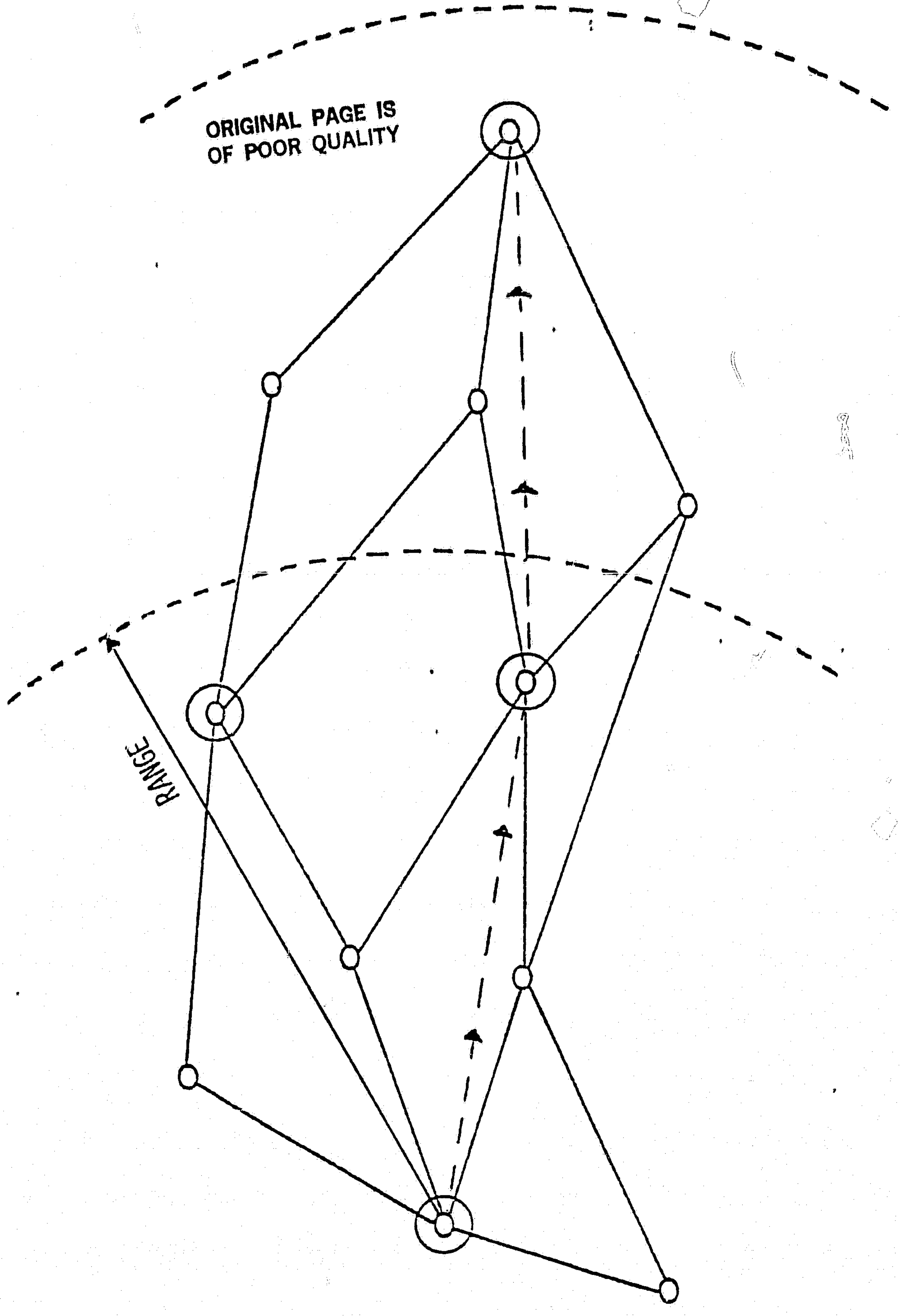


Figure 3 The route level representation

vocabulary. Figure 4 illustrates the transition from the route-level plan to the trajectory-level goals.

The trajectory level is below the route level in the conceptual level hierarchy because it depends upon the route generated by the route level. It requires the route produced by the route level to generate the actual trajectory goal. The route guides the planning at the trajectory. The route is the goal of the trajectory and the trajectory implements the route.

A flight segment is defined to be the takeoff airport, the sequence of airway segment between the takeoff airport and the landing airport, and the landing airport. The trajectory level divides a flight segment into three phases: the takeoff phase, the cruise phase, and the landing phase. The trajectory level generates the trajectory for each phase. For the cruise phase, the horizontal trajectory corresponds to the route. The aircraft performance knowledge base is an integral part of the trajectory level. Given the route and the goals of the aircraft airspeed, service ceiling, and range from the route level, the trajectory level checks the aircraft performance knowledge base to see if this can be accomplished. If this can not be done, the trajectory level suggests revisions to the route level and the route level will replan and generate another set of goals for the trajectory level.

For the instrument flight, the FAA has established required takeoff and landing trajectory for many airports [9,10]. The trajectory level uses these established trajectories as the trajectory goals for the takeoff phase and the landing phase. These trajectories are stored in the trajectory knowledge base and are retrieved as keyed by the route.

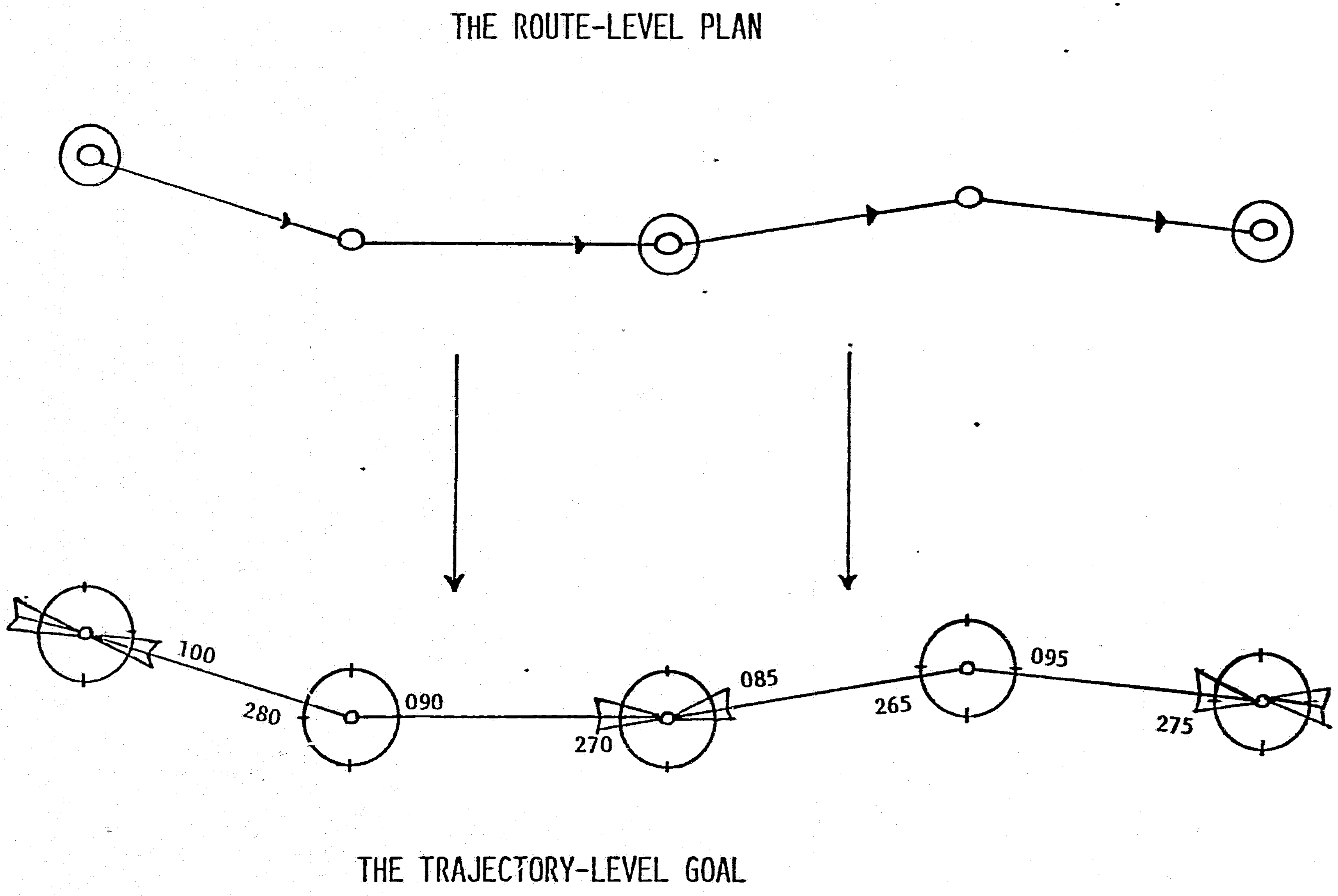


Figure 4 The semantic transformation between the levels

Figure 5 shows the mapping from the trajectory goals to the trajectory plan.

Another aspect of the trajectory level is navigation. The trajectory goals are the desired path of the aircraft. It specifies where the aircraft should be. It takes navigation to determines the aircraft location with respect to the desired aircraft flight path. It is also the responsibility of the trajectory level to determine the aircraft's location and the correction trajectory to rejoin the desired flight path should the aircraft wanders off the desired flight path.

3.2.3. The Flight Control Level

The flight control level is the conceptual level below the trajectory level. The flight control level is responsible for generating the plan to maneuver the flight controls to achieve a certain trajectory goal. The flight controls are the throttle, the fuel air mixture, the aeileron, the stabilator, the rudder, the flaps, and the landing gear. The aircraft is assumed to be the Piper Cherokee, a light, single engined aircraft. Larger commercial aircrafts have other additional flight controls. The trajectory goal is given by the trajectory level. The plan at the flight control level is a sequence of the flight control settings that achieves the given trajectory goal.

The flight control level is concerned with the aerodynamic knowledge. The aerodynamic knowledge include the forces that influences the flight trajectory. The aerodynamic knowledge base also includes the

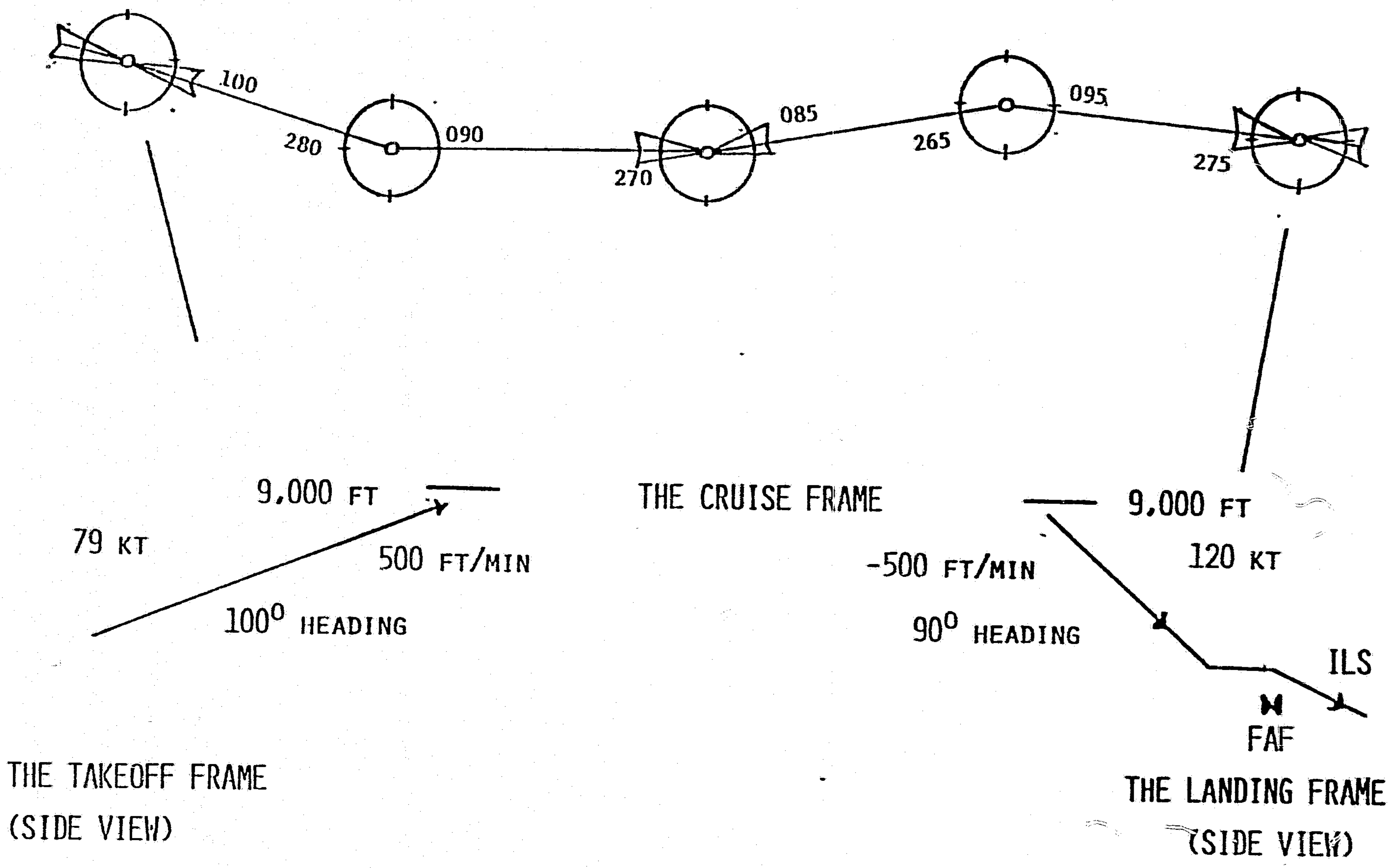


Figure 5 Planning at the trajectory level

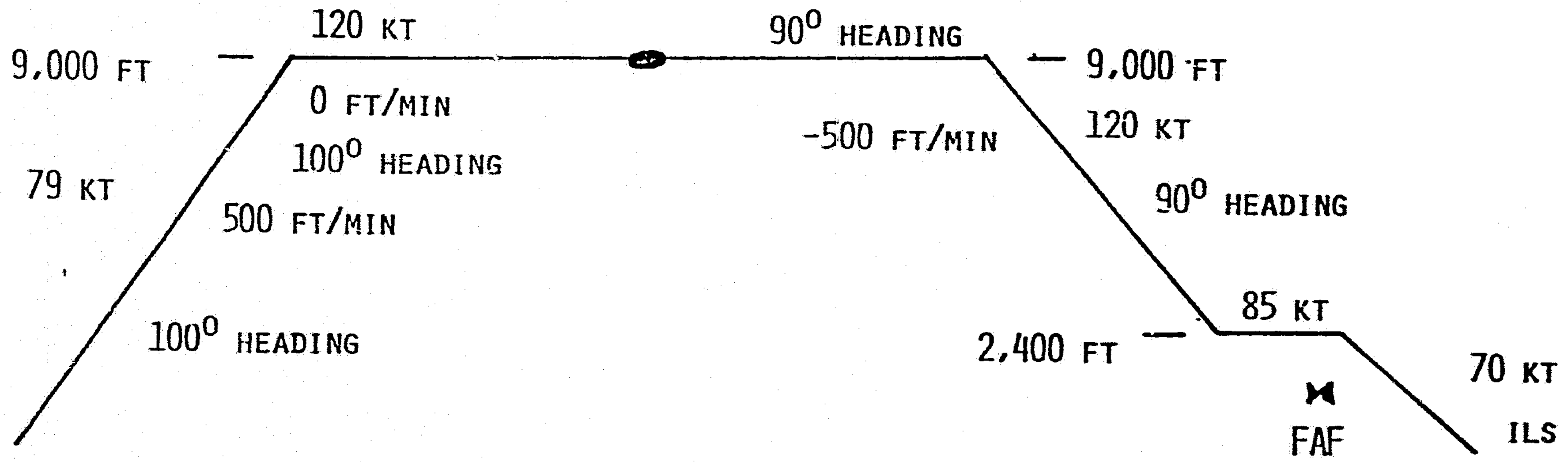
association between the flight controls and these forces. For example, the throttle is associated with the force of thrust, and for a given aircraft attitude, greater thrust results in greater climb rate. The stabilator controls the pitch attitude which in turn controls the airspeed. The flap increases the lift coefficient, thus enabling flight at lower airspeed. However, the flap also increases the drag coefficient, thus requiring more power to fly at lower airspeed. These are examples of the knowledge at the flight control level. The variables at the flight control level are tightly connected and interacting. Thus they form a causality framework. Figure 6 shows the mapping from the flight control level goals to the flight control level plan.

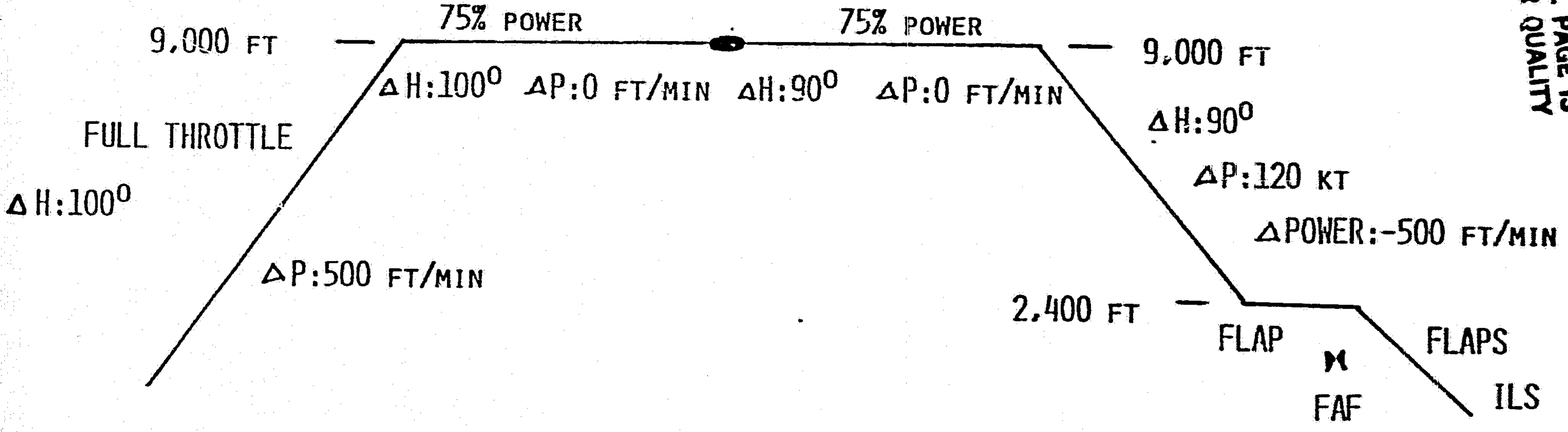
3.2.4. The Subsystems Level

The aircraft subsystems level is the conceptual level below both the trajectory level and the flight control level. The aircraft subsystems level performs the support role for both the trajectory level and the flight control level. The subsystems level generates plan to sustain the trajectory level by providing an uninterrupted electrical power to the navigational equipment. The subsystems level also generates plan to sustain the flight control level by ensuring a running engine. These are the subsystems support for our example aircraft, the Piper Cherokee. Larger commercial aircrafts would also have hydraulic and pneumatic support subsystems.

The relationship between the aircraft subsystems level and the two conceptual levels above it is an enablement relationship. This

THE FLIGHT-CONTROL-LEVEL GOAL





THE FLIGHT-CONTROL-LEVEL PLAN

enablement relationship is different from the form/function relationship between the other conceptual levels. In the form/function relationship, the form at the top level is implemented by the functions of the bottom level. In the enablement relationship, the bottom level enables the top level to achieve the top level's goal. For example, the planner can not navigate without powered navigational equipment. The proper throttle setting is useless if the engine died of fuel starvation.

The causality framework at the subsystems level is that of mechanical systems such as the electrical system and the fuel system. These systems are interacting. The electrical system powers the electrical fuel pump which sustains the engine. The engine then drives the alternator which powers the electrical system. A representation such as the Common Sense Algorithm can be designed to represent these mechanical systems [11,12,13]. Figure 7 illustrates the Common Sense Algorithm representation.

3.3. The Inter-level Dependencies

The causal framework determines the conceptual levels, and the planner at each level only has to consider the variables within the causal framework. This is because the variables within the causal framework are tightly related. The intra-level planning may include any of the planning techniques developed thus far, and possibly a recursive application of the levels architecture. The interesting feature of the conceptual levels planning architecture, however, is the inter-level relationships or dependencies.

ORIGINAL PAGE IS
OF POOR QUALITY

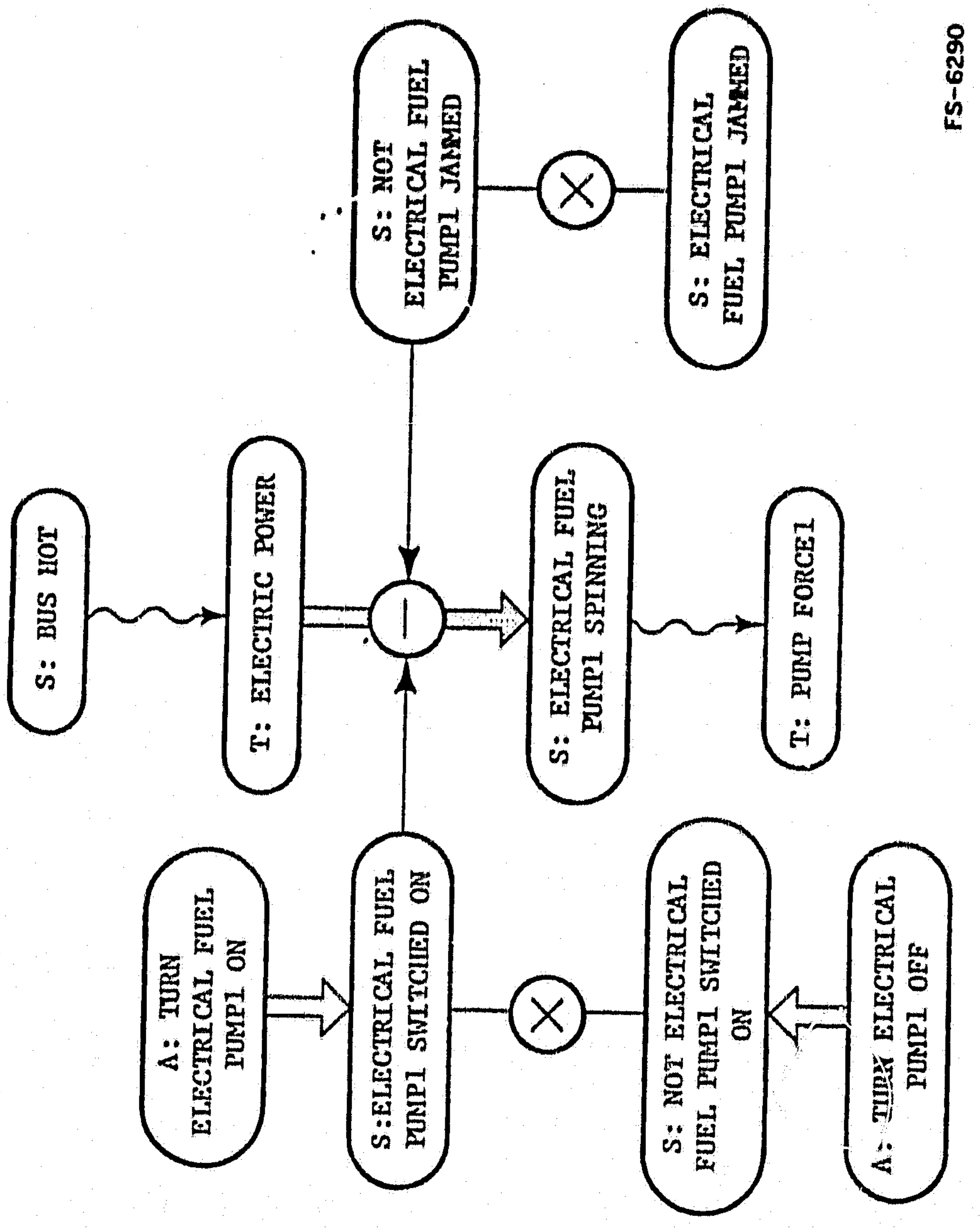

A: TURN ELECTRICAL FUEL PUMP1 ON
S:ELECTRICAL FUEL PUMP1 SWITCHED ON
S: NOT ELECTRICAL FUEL PUMP1 SWITCHED ON
A: ~~TURN~~ ELECTRICAL PUMP1 OFF
S: BUS HOT
T: ELECTRIC POWER
S: ELECTRICAL FUEL PUMP1 SPINNING
T: PUMP FORCE1
S: NOT ELECTRICAL FUEL PUMP1 JAMMED
S: ELECTRICAL FUEL PUMP1 JAMMED
FS-6290

Figure 7 The common sense algorithm representation

The previous section has already discussed in some detail the form/function and the precondition inter-level relationships. Both of these relationships are top-down in the sense that the top level gives the bottom level direction and guidance. Thus the direction of planning is top-down because the top level provides the necessary higher-level direction. Planning at the trajectory level first before planning at the route level would be in vain because the trajectory is probably in the wrong direction since the refueling airport has yet to be determined.

The inter-level dependency also operates in the bottom-up direction, though this is less obvious. In the case of the precondition inter-level relationship, the bottom level depends on the top level for the goal specification, but the top level also depends on the bottom level for the top-level operator capability. For example, the flight-control plan is void if the subsystems level can not keep the engine running. This kind of dependency continues in the form/function hierarchy in the bottom-up direction. This is because the upper level plan step is implicitly dependent on the lower level plan segment. The upper level plan step is implemented by a lower level plan segment, thus if the lower level plan segment can not deliver the expected result, then the upper plan step is invalid, thus invalidating the upper level plan.

The result of this bottom-up dependency is that the operator capabilities of the upper level depend on the lower level. Thus the operator capabilities at the upper level may change due to changes at the lower levels. For example, suppose consistent plans have been completed at all four levels. Then the engine runs hot and the subsystems planner wants to cut power by 40%. This reduces the throttle setting at the flight-control level, which reduces the airspeed and the altitude ceiling at the trajectory level, which invalidates an airway segment at the route level because there is a mountain under the airway segment. Thus a change at the lowest level affects even the highest level.

3.4. Inter-level Semantic Transformation

Since the inter-level dependencies run both up and down the conceptual levels hierarchy, the levels must communicate with each other. Communication is not straightforward since, by design, the levels do not have to speak with the same vocabulary. In the top-down direction, the upper level specifies the goal for the lower level. A completed plan at the upper level becomes the lower level's goal. Semantic transformation is necessary to make the demand comprehensible. The same is true in the reverse direction. The lower level specifies the upper level's operator capability. A dead engine at the subsystems level is translated to effectively zero throttle capability at the flight-control level and then zero climb rate capability at the trajectory level, etc. Thus, semantic transformation knowledge base is necessary at each level for communication in both directions.

3.5. Levels Summary

The conceptual levels planning architecture is a semantic partitioning and organization of the domain knowledge. The partitioning divides the domain world into smaller fiefdoms. The planner within a partition can concentrate on its own fief and ignore the rest of the world. The organization specifies the relationships between the fiefs and makes the partitions meaningful. A random partitioning is senseless because it has no organization.

A unique feature of the conceptual levels architecture is that there is planning consistency within a level and there is also planning consistency over the levels hierarchy. The planner in each level makes sure the plan within each level is true with respect to the factors inside the level. The plan within each level is also true to the factors outside each level. This is accomplished by inter-level communication. Planning direction is passed from the top down. Operator capability is passed from the bottom up. Thus the planner considers not only the factors within its own level directly, but it also considers the factors outside its level in a more indirect fashion.

Unlike previous planning systems, the conceptual levels architecture defines uniform levels of domain semantics. The plans at each level all makes sense with respect to their own level (context). Thus a complete plan at each level can be constructed, and a complete plan over the levels hierarchy consists of a complete plan at each level and the plans are consistent with each other.

The uniform levels of domain semantics definition enables the focusing of attention. The planner within a level can almost ignore the rest of the world. The levels hierarchy also specifies where to focus the attention next. This is covered in more detail in the following section. The nearly independent levels can also support different knowledge representation at each level. Since knowledge representation should be fitted to the need and since the level semantics may be different, there can be a mixture of knowledge representations in the levels hierarchy.

The form/function and precondition inter-level dependencies allow the vertical decomposition of a task. The divide-and-conquer paradigm advocates the decomposition of a task. However, in actual usage, the divide-and-conquer paradigm provides the horizontal subtask decomposition, or subtasks of similar semantics. The conceptual levels hierarchy specifies the vertical subtask decomposition where the vertical decomposition indicates the subtasks' semantics are different across the form/function or precondition dependencies. These inter-level dependencies also enables higher-level planning direction. Plan consistency over the entire hierarchy starts at the top level. When the top-level plan is completed, it is passed downward as the goal for the lower level, etc.

The conceptual levels hierarchy also enables partial planning where planning does not have to proceed down to the last detail. For example, as long as the route level and the trajectory level have satisfactory plans and the subsystems level can provide the support, the planning at the flight-control level can be mostly ignored except for the immediate future.

The conceptual levels hierarchy provides the theoretical foundation for a new approach to planning. The hierarchy alone, however, is not a planning system. In addition to the hierarchy, a planning control mechanism is required. The planning control mechanism for the levels architecture will be covered in the next section.

4. The Planning Control Mechanism

The conceptual levels hierarchy specifies complex relationships within and without a level. Such complex relationships require a sophisticated planning control mechanism. Planning activities in the conceptual levels hierarchy can be broken down to intra-level planning activities and inter-level planning activities.

4.1. Intra-level Planning

The intra-level activities consist of the plan generation process once the goal is given. Of course, in this case, the goals are obtained through the inter-level planning activities. The intra-level planning activities occur inside a conceptual level. Within the route level, the intra-level planning process generates a route from the origin airport to the destination airport that satisfies the constraints applicable to the route. Within the trajectory level, the intra-level planning process generates a trajectory that implements the route and also satisfy the applicable trajectory constraints such as controlled airspace and the aircraft performance limitations. When the trajectory is worked out, the flight control level planner plans the control actions that will achieve the desired trajectory.

The intra-level planning activities generates the plan at a particular conceptual level. Because of the conceptual levels architecture, the planner at a given level only has to consider the variables at that particular level. Thus the size of the problem is reduced from the entire flight domain to the size of that conceptual level. This reduction is the power of the architecture.

While the intra-level planner has only to examine a subset of the domain, someone else has to make sure the total picture is coherent and consistent. Some mechanism has to maintain the overall viewpoint to make sure all the subplans add up to a functional total plan. This is the responsibility of the inter-level plan control mechanism. While the conceptual levels architecture enables decomposition, the inter-level plan control mechanism enables the integration of the pieces.

4.2. Inter-level Planning

The inter-level controls can be classified into two aspects: focusing on a level and transitioning the levels interface. Transitioning the levels interface is not interesting; it is merely shifting the focus up or down one level. However, the reason for the focus shift is interesting. Focus means the narrowing of the scope. Focussing the attention has meant in previous works the current locus of planning activities. For example, the planner may be searching for the operators that can achieve a goal or the planner may be contemplating the decomposition of a goal. If the planner goes to another part of the plan to contemplate other problems, the planner is said to have changed its focus. The focussing of attention in the context of the conceptual levels architecture has a different meaning. In this context, focussing means limiting the scope to a conceptual level. In the conceptual levels architecture, the focus shifts frequently as the inter-level plan control mechanism enforces coherence over the entire hierarchy.

The inter-level planning control mechanism has precedence over the intra-level planners and controls the intra-level planners. The inter-level planning control mechanism is rooted in the inter-level relationships. The form/function inter-level relationship results in both top-down and bottom-up control actions. The precondition inter-level relationship results in bottom-up control actions.

Control proceeds top-down when a plan in the top level is passed to the lower level as the desired goal. For example, when the plan at the route level is completed, the route is passed to the trajectory level as the trajectory level goals. Then the focus is shifted to the trajectory level as the trajectory level planner plans to achieve the route. Control also flows bottom-up because the lower-level defines the top-level operator capabilities. For example, if the subsystems level can not maintains engine operation, then the operators at the flight control level become invalid. Thus if changes occur at the lower level, the focus will shift to the upper level to verify that the upper-level plan is still valid.

The reasons for making a focus shift can be due to the PROPAGATE-VALUE-UP, the PROPAGATE-PLAN-DOWN, the PROPAGATE-VALUE-REQUEST-DOWN, and the PROPAGATE-GOAL-REQUEST-UP actions. The PROPAGATE-VALUE-UP action is used to communicate to the upper level its operator capabilities. The PROPAGATE-PLAN-DOWN is used when the upper level has completed its plan and wishes to pass it down as the goal for the lower level. The PROPAGATE-VALUE-REQUEST-DOWN action is used when the upper level requests a clarification of its operator capabilities. The PROPAGATE-GOAL-REQUEST-UP action is used when a lower level requests a clarification of its goals from the upper level.

When attention is first focused on a level, the control mechanism needs to determine what needs to be done, or what caused the focusing of attention on this level? There are many possible causes to the focusing of attention on a level. Whatever the causes, the main actions at a level are propagating a message, plan at that level, and recovery plan at that level. Plan at that level results in the PLAN action which calls the planner for that level. PLAN can be described as:

```
IF (NOT HAS GOAL) THEN PROPAGATE-GOAL-REQUEST-UP
IF (NOT HAS OPERATORS) THEN PROPAGATE-VALUE-REQUEST-DOWN
CALL PLANNER
```

Recovery plan at that level results in the action RECOVERY-PLAN which differs from PLAN in that RECOVERY-PLAN remedies small perturbations. RECOVERY-PLAN can be described as:

```
LOCATE-PERTURBATION
PLAN
PATCH-PLAN
```

The action taken when first focused on a level depends on the cause of shifting the focus to that level. If the cause is because a value is requested from above, the action is:

```
IF (VALUES REQUESTED) THEN
  IF (HAS VALUE) THEN PROPAGATE-VALUE-UP
    ELSE PROPAGATE-VALUE-UP(PROPAGATE-VALUE-REQUEST-DOWN)
```

The other actions are:

```
IF (GOAL REQUESTED) THEN
  PLAN
  PROGAPAGE-PLAN-DOWN
```

```
IF (SUPPORT VARIABLES CHANGED) THEN
  RECOVERY-PLAN
  PROPAGATE-VALUE-UP
  PROPAGATE-PLAN-DOWN
```

```
IF (PLAN DEVIATION OCCURRED) THEN
  RECOVERY-PLAN
  PROPAGATE-VALUE-UP
  PROPAGATE-PLAN-DOWN
```

```
IF (NEW GOAL OR ADJUSTED GOAL) THEN
  PLAN
  PROPAGATE-PLAN-DOWN
```

In addition to these inter-level actions, there are also three other actions that starts the ball rolling. The START-AT-THE-TOP-LEVEL action starts the planning at the top level in the beginning. The LOCATE-LEVEL action locates the appropriate level for repair work when either a support has changed or when the aircraft has drifted away from the original plan. The ORDER-BY-PRIORITY action determines the priority when several disturbances require attention.

The above actions describe the planning control actions necessary to support planning over the conceptual levels planning architecture. The focus of this planning control mechanism research is on the activities due to inter-level relationships.

5. Summary

The conceptual levels planning architecture is unique because it uses the semantic organization of the domain knowledge to achieve higher-level planning direction. This approach is motivated by the causal framework observation that some variables are more tightly related than others. A tightly related group of variables forms a conceptual level. A planner within the level plans directly with the factors within the level and indirectly with factors outside the level. The factors from outside the level arrive via inter-level messages. Semantic transformation may be necessary to communicate across the level boundaries.

The inter-level planning control mechanism has precedence over the intra-level planners and controls the intra-level planners. The inter-level planning control mechanism is rooted in the inter-level relationships. The form/function and precondition inter-level relationships give the levels architecture its power. These two kinds of inter-level relationship enable the high-level planning that guides the lower-level planning. The organization of domain knowledge by the form/function and the precondition relationships coupled with the levels planning control mechanism give the conceptual levels architecture its power.

The conceptual levels architecture enables the focusing of attention on a small portion of the domain and the focusing of attention on a level of the planning process. The levels hierarchy also enables the vertical decomposition of a task because the hierarchy enables a vertical definition of the domain semantics. The levels architecture provides higher-level planning direction since the completed higher-level plan becomes the goal for the lower level. The levels architecture supports non-homogenous knowledge representation. This is because planning at each level is buffered. Lastly, the levels architecture enables partial planning. Again, this falls out from the vertical definition of domain semantics.

The work accomplished thus far consists of the design of a semantically oriented planning architecture. Previous approaches to complexity control have been more syntactically oriented than semantically oriented. The conceptual levels approach organizes the domain knowledge into levels that are based on the form/function and the precondition inter-level relationships. This architecture has been applied to the aircraft flight domain and a walk-through scenario is easily constructed. Lastly, an initial design of the inter-level planning-control mechanism has been done. This mechanism performs meta-planning in the levels context.

6. References

1. R.T. Chien, et al, "An Intelligent System for Monitoring and Diagnosis in Cookpit Environment, TR-3 on Artificial Intelligence and Human Error Prevention: A Computer-Aided Decision Making Approach," T-79, Coordinated Science Laboratory, University of Illinois, Urbana, Illinois (January 1980).

2. A. Tate, "Interacting Goals and their Use," Proceedings of the Fourth International Joint Conference on Artificial Intelligence, pp. 215-128 (September 3-8, 1975).

3. R. Waldinger, "Achieving Several Goals Simultaneously," pp. 94-136 in Machine Intelligence 8, ed. D. Michie, Ellis Horwood Limited, Chichester, England (1977).

4. E. D. Sacerdoti, "Planning in a Hierarchy of Abstraction Spaces," Artificial Intelligence, Vol. 5, (2) pp. 115-135 ().

5. E. D. Sacerdoti, A Structure for Plans and Behavior, Elsevier, New York, NY (1977).

6. E. D. Sacerdoti, "The Non-Linear Nature of Plans," Proceedings of the Fourth International Joint Conference on Artificial Intelligence, pp. 206-214 (September 3-8, 1975).

7. P. E. Friedland, "Knowledge-based Experiment Design in Molecular Genetics," Stanford Heuristic Programming Project Memo HPP-79-29, Department of Computer Science, Stanford University, Stanford, CA (August 1979).

8. Mark Stefik, "Planning with Constraints," Memo HPP-80-2, Computer Science Department, Stanford University, Stanford, CA (January

1980).

9. , <u>Airman's Information Manual</u>, Aero Publishers, Inc., Fullbrook CA (1981).

10. Department of Transportation, <u>Instrument Flying Handbook</u>, Federal Aviation Administration (1980).

11. C. Rieger, "An Organization of Knowledge for Problem Solving and Language Comprehension," <u>Artificial Intelligence</u>, Vol. 7, pp. 89-127 (1976).

12. C. Rieger, "The Representation and Selection of Commonsense Knowledge for Natural Language Comprehension," TR-458, Department of Computer Science, University of Maryland, College Park, Maryland (May 1976).

13. C. Rieger and M. Grinberg, "The Causal Representation and Simulation of Physical Mechanisms," TR-495, Department of Computer Science, University of Maryland, College Park, MD (1976).

Chapter III

Model-Based Diagnosis

1. Overview

During the past year, our research is focused on finding a suitable way to model the aircraft mechanism to provide the knowledge base for the rationalization of failure possibilities.

Our previous research [1] has resulted in a verification method for "given" failure assertions. With this method, a fault-asserted mechanism is viewed as a "new" mechanism. The verification process is proceeded in following two phases: model-reconstruction and measurement-propagation. In the first phase, the constraint model for the failure-asserted mechanism is established by modifying the constraint descriptions of fault-asserted component(s). In the second phase, we use the new constraint model to analyze sensory measurements. The specific technique involved is called "constraint propagation" which has also been addressed by other artificial intelligence researches [2,3,4]. Our contributions are on the generalization of qualitative modelings and their interpretations which enables us to describe some quantitatively imprecise, yet useful, engineering knowledge. The result of constraint analysis can be one of following two cases: (1) sensory measurements are propagated through the new constraint model without any conflict, or (2) at least one conflict is detected during the propagation process. In the former case, the underlying failure assertion is accepted as a possibility, and is justified by a set of inferred parameters. In the later case, the failure assertion is rejected since it fails to consistently explain all sensory measurements.

With the establishment of verification process, we can objectively evaluate a heuristically-infered failure assertion. To complete our theory of diagnosis, we need to develop a reasoning process to infer from the mechanism model a set of failure hypotheses by which deviated measurements can be explained. This report discusses results summarized from our research on this direction, which includes following topics: (1) how to model the functional behavior of the mechanism, and (2) how to reason with the mechanism model to assert failure hypotheses.

2. Related Works

Existent artificial intelligence works in the area of diagnoses are based on two basic approaches: the production-rule-based expert approach and the mechanism-model-based approach. Although intended domains of these researches may not be exactly airplane mechanisms, we will discuss problems involved in generalizations of these approaches to our domain of interest.

2.1. Rule-based Expert Approach

The production-system paradigm [5] has been implemented for various applications: MYCIN [6] for medical diagnoses, PROSPECTOR [7] for mineral exploration, SACON [8] for structural analysis, and SU/X [9] for signal interpretation. Although none of above works are directly addressed to mechanism diagnoses, its basic scheme, as provided by EMYCIN [10], can be readily applied to build a rule-based mechanism diagnosis system. Following such rule-based approach, however, the computer knowledge base encodes nothing but diagnostic rules resulted from human-experts' interpretations of their mechanism understandings. Since the computer itself has no understanding of the objective mechanism, any modification of rules requires intervention of human experts [11].

The weakness of rule-based approach thus is clear: the experience gained from building an expert system for a specific mechanism is "wasted" in the sense that it can not be transferred into another mechanism in the same domain. The remedy requires a fundamentally different approach to build a diagnosis expert system. The computer is programmed to use its understanding models of the mechanism, as encoded in the knowledge base, to perform diagnoses. Following such approach, the experience accumulated from building models for a specific mechanism can help to build diagnosis system for other mechanism in the same domain. Next, we will discuss two instances of diagnosis systems based on such model-based approach.

2.2. Model-based Diagnosis Approach

In the area of model-based diagnoses, we discuss two MIT works based on rather different modeling schemes. In the first instance, Brown demonstrates that troubleshootings can be based on the hierarchical design-plan of a radio receiver. Thus, the knowledge base encodes "global" understandings of a mechanism. In the second instance, deKleer use only the constraint model at component level to perform diagnoses on electronic circuits. Any use of "teleology" (or global knowledge) about the mechanism is explicitly excluded.

WATSON [12] is a computer program to perform troubleshootings on radio-receivers. Brown's diagnosis strategy is to backtrace faulty outputs among "stages" as defined by the hierarchical design-plan of a radio-receiver. The objective is to localize a possible faulty component with least measurements.

WATSON's diagnosis strategy is not applicable to our monitoring-diagnosis tasks for two basic reasons:

(1) WATSON's intended environments allow selections of test-points, injections of experimental signals, and physical separations of components. All these "diagnosis-initiated" requests are not permitted in our the airplane environment where only information available are measurements from pre-installed sensors. Brown's assumed enviornments make it unnecessary for the troubleshooting strategy to involve in complicated parallel hypothesis generations and evaluations which are essential for diagnoses in our airplane environments.

(2) Brown's mechanism model is based on the original design-plan which does not always meet implicit assumptions for his causality-based diagnosis strategy. A better alternative will be to develop a consistent modeling scheme which can result in a mechanism model suitable for diagnosis reasoning. We will further pursuit this subject later in the discussion of our approach.

In his work for localizing faults in electronic circuits [13], deKleer pursuits a purely local method for diagnosis. Constraint models for circuit components are explicitly linked together by the circuit topology to form the model for the overall circuit. Given measurements are propagated through the constraint model of the circuit to deduce new parametrical information. The diagnostic strategy is based on "coincidence" which occurs when one circuit parameter can be deduced in several different ways. When a contradiction is detected at a coincidence, deKleer's program looks back to all components involved in the deduction of that parameter and logically infers a set of possibly faulted components.

The major weakness of deKleer's local approach lies in its inability to incorporate global understanding of the circuit. More specifically, it fails to utilize normal measurements of a no-fault mechanism as an important information source for diagnoses. Also, the lack of knowledge on the functional structure of the mechanism severely limits its ability to propagate, thus to use, given data. Our theory, as discussed below, will show that by resorting to the global functional understanding of a mechanism, we can make better use of given measurements.

3. Our Diagnosis Approach

Our theory of diagnosis is based on a "hypothesization-verification" paradigm which has also adopted by many other artificial intelligence systems [14,15]. For mechanism diagnoses, the challenging issue is to implement such paradigm based on models of the mechanism. Other model-based approaches are based their diagnosis strategies on a single perspective of the mechanism (such as Brown's using of design-plan model and deKleer's using of constraint model) which often fail to take full advantage of measurements available. Our approach first assumes that there are more than one point of view to model a mechanism. Our previous research results in a verification theory based on the constraint model which describes the mechanism from a analytical point of view. We now discuss our progress in another direction, namely, the modeling of mechanism from a functional point of view and the use of such model to rationalize failure assertions.

3.1. The Functional Model

In contrast to an analytical perspective which views the behavior of a mechanism as an equilibrium state satisfying all constraints of its components, the functional perspective recognizes that there are interactions among components.

We characterize the "interaction" between two components as a flow of certain "medium" which can be "fluid" type (such as fuel, oil), or "energy" type (such as heat, electricity, or torque). Based on such "flow" interpretation, we build up the functional description of the mechanism at component level, namely, components are acting as basic functional units which "receive" and "deliver" flows.

As an example in figure 1, a fuel-delivery mechanism driven electrically is pumping fuel to the engine. The flow description identifies all the meaningful flow interactions among components of the mechanism, as shown in figure 2.

In following sections, we discuss the concept of "subsystem" which we impose on the basic flow description of the mechanism. Then causalities among subsystem are studied, which results in the causal-dependency description of a mechanism.

3.1.1. The Subsystems

3.1.1.1. The Development of Subsystem Concept

Based on what a functional unit does with its flows, we further classify functional components into two categories: for those who merely "pass" or "consume" a particular medium flow, we call them "passive" (as analogous to "passive components" in electronic circuits), and for those who either "generate" flows or "convert" one medium flow to the other, we call them "active" (again, analogous to "active components" in electronic circuits). For example, in figure 2, the component "electrical source" is active component which "generates" electrical flow, so is "fuel pump" which "converts" electrical flow into "fuel flow". L1, filter, L2, and nozzle are passive because they either pass fuel flows or consumes fuel flow.

Now we impose a functional organization on top of the basic flow description based on the identifications of active functional units. We group an active component with the set of passive components which are "driven" by the generated medium flow, and call the whole functional group a "subsystem". As shown in

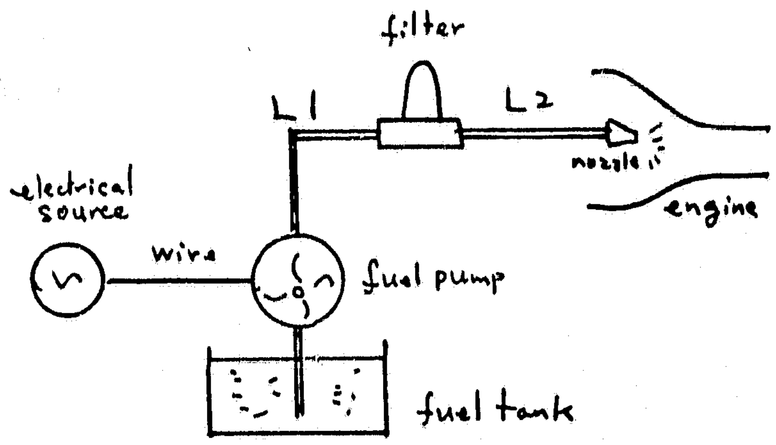


Figure 1. A Physical Description of the Fuel-Delivery Mechanism.

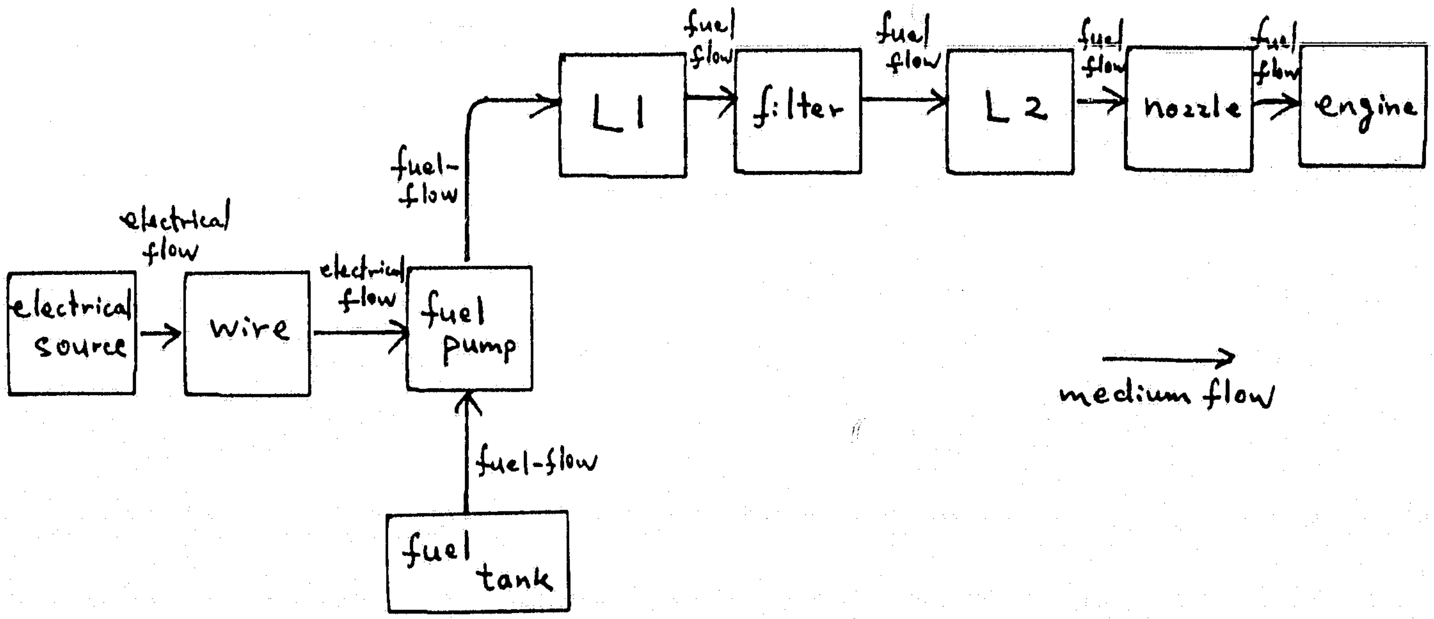


Figure 2. Flow Description of the Fuel-Delivery Mechanism.

figure 3, we group the fuel-delivery mechanism in two subsystems, called electrical subsystem and fuel subsystem respectively.

We thus identify two essential actions underlying a subsystem concept: "driving" and "response". "Driving" is initiated by the active component which under proper enablement create a tendency to cause medium flows among passive components of the subsystem. "Response" is the medium flow in the environment of passive components resulting from the driving action. Within the environment of response, variables (or functional parameters) which characterize medium flows among passive components are associated with a set of physical laws related to the physics nature of the medium involved.

For example in the fuel subsystem of figure 3, the running of fuel pump with the fuel supply of a non-empty fuel pump creates a driving tendency to cause fuel flow in the environment formed by its passive components, namely L1, filter, L2, nozzle, and engine. The variables P's and F's characterize the fuel flow among the environment, which are governed by the laws of fluid-dynamics.

3.1.1.2. Frame Representation for Subsystems

A frame-like representation [16] is chosen to provide "slots" for the description of knowledge surrounding a subsystem. Each essential aspect of a particular subsystem is to be filled under its corresponding slot, as listed below. Since all

ORIGINAL PAGE IS
OF POOR QUALITY

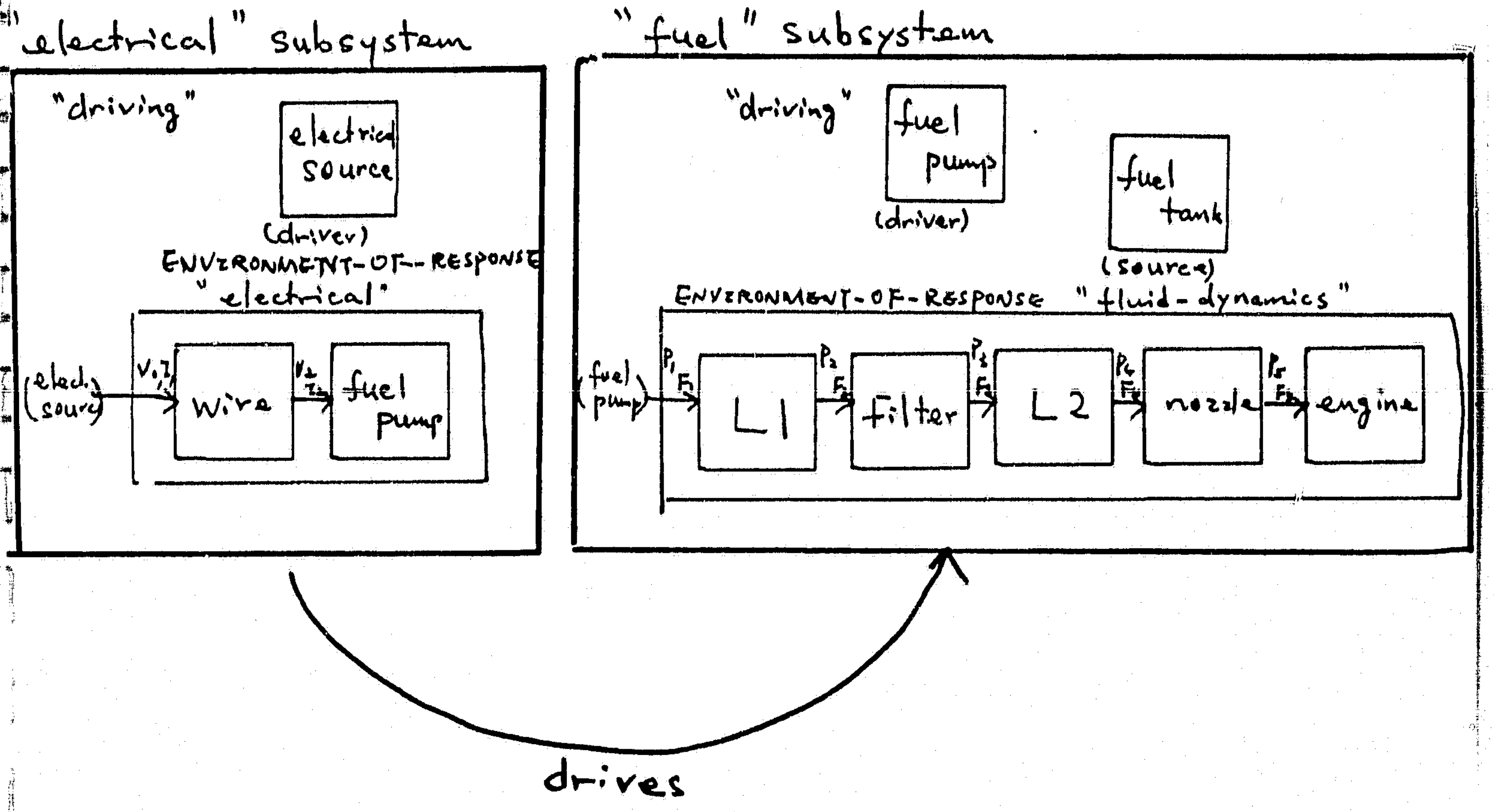


Figure 3. Subsystem Identifications of Fuel-Delivery Mechanism.

physical components grouped under a subsystem are explicitly accounted for and assigned to their functional roles, the subsystem frame serves as a conceptual linkage between the physical structure and the functional description of a mechanism.

(1) MEDIUM -

The type of medium which underlies functional interactions among components of this subsystem. Components within a subsystem encounter a complete cycle of the medium, from its source/generation to its drain/consumption.

(2) DRIVER -

The component identified as the "driver" (or active component). Under proper enablement, as to be specified within this slot, the driving action causes medium flows among "passive" components, as specified by the environment slot, of the subsystem.

(3) SOURCE -

When the medium is of type "fluid", the source of medium (such as the fuel tank or the oil reservoir) is explicit specified. when energy type of medium is involved, the medium is always generated from the DRIVER, thus the SOURCE is the same as the DRIVER.

(4) ENVIRONMENT-OF-RESPONSE -

Passive components which react to the driving tendency of the DRIVER are specified. The environment are based on the flow organization of passive components (i.e., cascaded or parallel). The sub-slot "parameter-definition" links functional parameters of this environment to locations of its physical structure. Additionally, a sub-slot "boundary-conditions" interfaces the environment to its driver. A sub-slot called "applied-laws" refers to the set of physical laws applicable to this environment. These laws provide parameters of this environment with proper interpretations.

Having defined various aspects of a subsystem, we now show in figure 4 a subsystem frame which describes the fuel-subsystem in figure 3.

```
SUBSYSTEM-FRAME:

        NAME: fuel-subsystem
        MEDIUM: (fuel (is-a fluid))

        DRIVER: (fuel-pump
                        (enabled-when
                            (fuel-pump running (> RPM 2400))
                            (fuel-tank not-empty (> quantity 0))
                        )
                )

        SOURCE: (fuel-tank
                        (capacity 5000)
                        (quantity (if-needed (sensor-reading Q)))

        ENVIRONMENT-OF-RESPONSE:
                (passive-components (L1 filter L2 nozzle engine))
                (path-structure (cascaded L1 filter L2 nozzle engine))
                (boundary-condition (connect fuel-pump L1))
                (parameter-definition
                        (flow-to fuel-pump L1 (P1 F1))
                        (flow-to L1 filter (P2 F2))
                        (flow-to filter L2 (P3 F3))
                        (flow-to L2 nozzle (P4 F4))
                        (flow-to nozzle engine (P5 F5))
                )
                (applied-laws "fluid-dynamics")
```

Figure 4. A frame representation for the fuel-subsystem.

3.1.2. Subsystem Dependencies

A subsystem groups a set of component with a particular functional perspective. As a result, variables of the mechanism are partitioned accordingly. Two subsystems interact when they share same component(s), i.e., at least one component is playing dual functional roles in both subsystems. Thus a causal relationship can be imposed among subsystems. If a component X is a passive member of environment-of-response in subsystem A also acts as the driver in another subsystem B, then we say that subsystem A "drives" subsystem B, meaning that if both subsystems A and B are not working properly, it is likely that B's problem may be caused by A's. This "driving-driven" relationship can be further explained as following: Component X works as a passive member in the response environment of subsystem A, which because of the driving action in A "passes" or "receives" the medium flow of subsystem A. As a result, it enables X to work as the driver in subsystem B, which in turns cause the response in subsystem B. The set of variables associated with B is thus causally related to the set of variables associated with A.

We again use the example in figure 3 to illustrate this point: Components "electrical source", "wire", and "fuel-pump" are grouped under "electrical subsystem", which models the electrical side of the mechanism. Voltage and current parameters are thus associated with electrical subsystem, which are governed by the laws of electricity. Similarly, "fuel-pump", "fuel-tank", "L1", "filter", "L2", "nozzle", and "engine" are grouped under fuel-subsystem, which applied laws of fluid-dynamics to describes the relationship among various fuel-flow parameters (P's and F's). In this mechanism, the fuel-pump plays dual functional roles in both subsystems. In electrical subsystem, it work as a passive load, which passively receives electrical flow. As the result, the fuel-pump runs, which enables it to act as the driver in the fuel-subsystem to cause fuel flows among passive components of the fuel subsystem.

In a more complicated mechanism, the causal dependencies among its subsystems can be generally described as an "AND-OR" graph of subsystems. For example, figure 5 shows the subsystem dependencies of a DC-10 like airplane. With extensive "redundant" arrangement, the fuel-subsystem is driven by either of the electrical buses. The electrical subsystem in turn is driven by either engine, which also drives its corresponding oil-subsystem and hydraulic-subsystem.

3.1.3. Conclusion

(1) Our functional model provide two levels of functional description of a mechanism. At the component level, it treats each component as a functional unit which interacts with other functional units via medium flows. At subsystem level, it describes causal dependencies among subsystems in terms of "driving-driven" relationships, which result in an "AND-OR" graph of subsystems.

(2) Each subsystem takes a particular functional perspective to group components. Thus, variables of the mechanism are divided into meaningful functional groups at subsystem level.

(3) Causal-dependency relationships at subystem level are explicitly specified which form the basis for fault isolation process.

ORIGINAL PAGE IS
OF POOR QUALITY

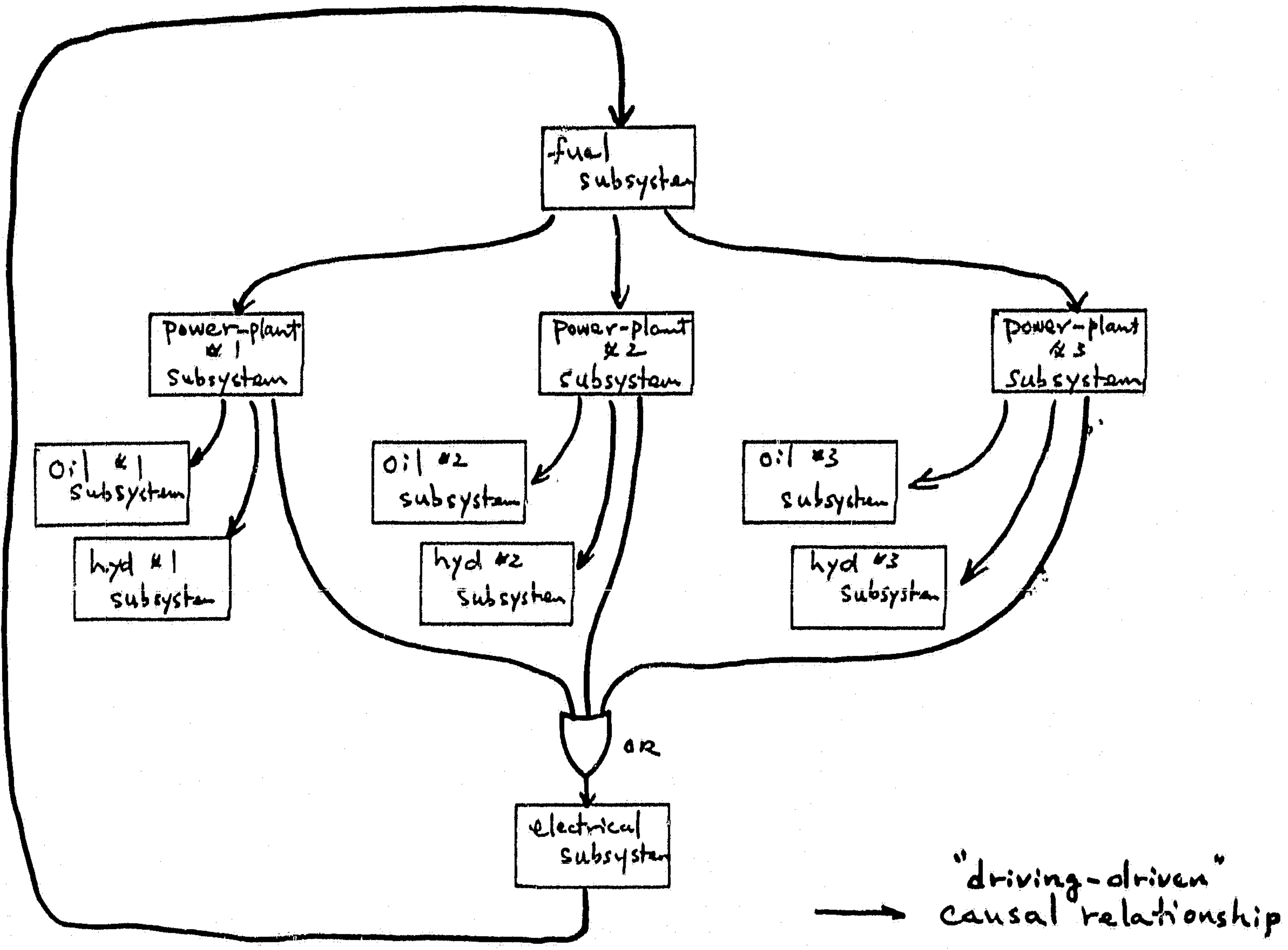


Figure 5. Causal Dependencies among Subsystems of a DC-10 Airplane.

3.2. Diagnostic Inference

In this section we discuss the inference strategy to heuristically generate failure assertion using our flow model. The process of failure assertion is consisted of two sub-strategies, fault isolation and fault hypothesization. We will discuss each process in detail in following sections.

3.2.1. Fault Isolation

The fault-isolation strategy uses subsystem causal dependencies relationships to heuristically isolate failure within a particular subsystem. Since mechanism variables are partitioned by subsystems, isolation of subsystem also means focusing on a particular set of variables.

The isolation heuristics is derived from "driving-driven" relationships of the functional model. The relationship says, if subsystem A drives subsystem B and symptoms are detected at both subsystem, A is more likely to fail then B. A heuristical backtracing strategy readily follows. When a deviated measurement is detected in a subsystem, we backtrace the causal-dependency link to look for possible symptoms in other subsystems. If another abnormal subsystem exists during the causal backtracing process, we will switch our focus on that subsystem. The process ends when we detect (1) a normally-functioning subsystem, or (2) a subsystem which does not drive by other subsystem. The result of causal backtracing process is a failure propagation trace of subsytems which describes the possible path of failure propagation. This failure propagation trace will guide the fault hypothesization process, which will be discuss in next section.

Use the fuel-delivery mechanism is figure 3 as an example. If symptom is detected in the fuel subsystem, (P3 low) for example, the following isolation reasoning follows:

(1) If fuel-pump is know to be running normally, i.e., RPM > 2400, the hypothesization strategy will be applied to fuel subsystem.

(2) If the fuel-pump RPM is either below 2400 or unknown, the isolation strategy will backtrace and focus on the variables of electrical subsystem.

In a general case as shown in figure 6, the isolation strategy will enable us to associate symptom in subsystem A with symptom in subsystem D, thus avoid detailed analyses on less-likely subsystems B, C, E, F, and G.

3.2.2. Fault Hypothesization

Based on the failure propagation trace resulted from the isolation strategy, the hypothesization process focus on the most-likely faulty subsystem. The interpretation of variables in the subsystem is provided by the set of physical laws which governs the environment of response. For example, the symptom (V2 low) and (I2 low) in figure 3 will lead to following fault assertions:

(1) (fuel-pump (resistance low))

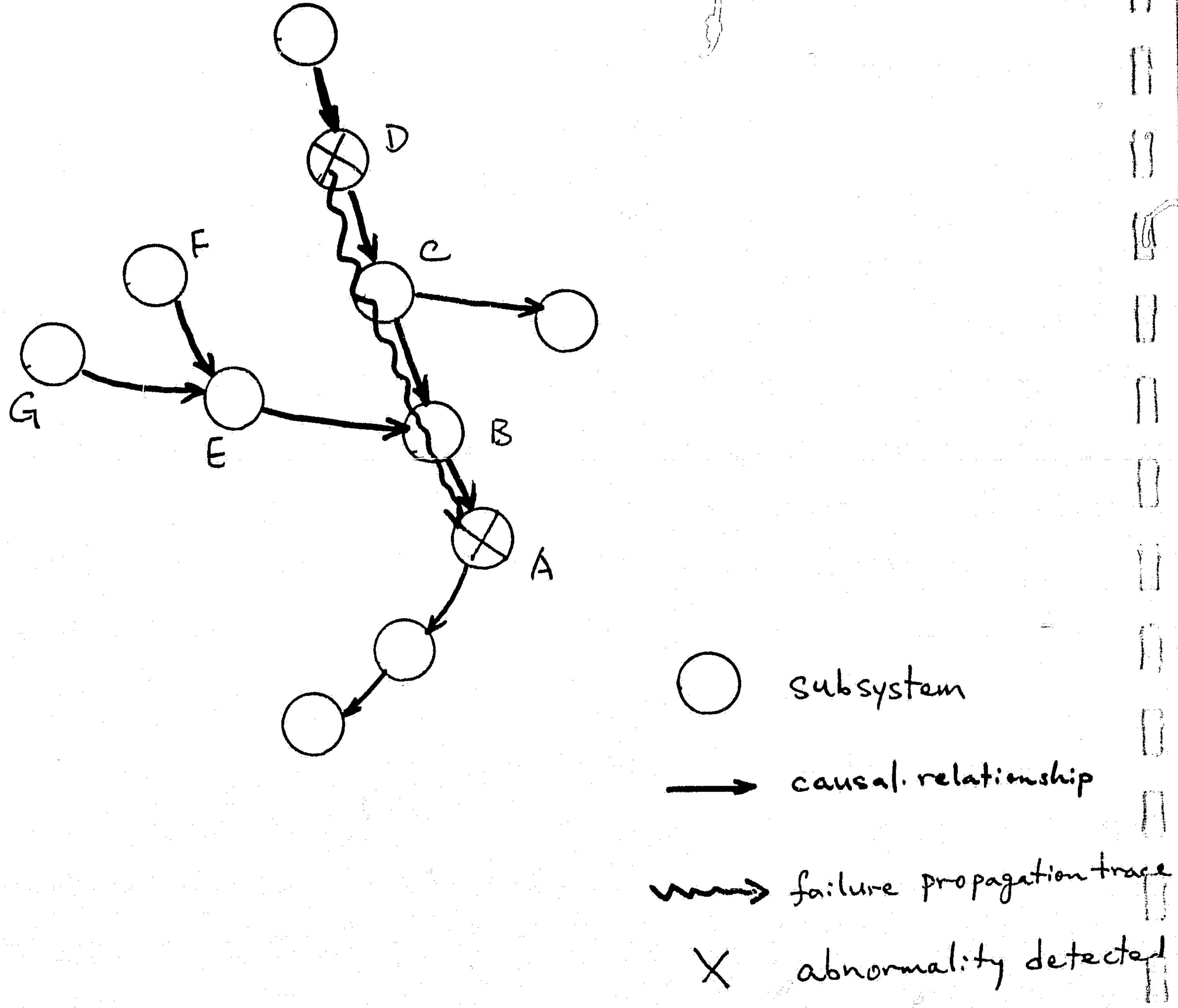


Figure 6. Fault Isolation through Subsystem Causal-Dependencies.

(2) (wire (resistance high))

(3) (electrical-source (voltage high))

These three hypotheses based on analyses of electrical variables will be verified by the constraint verification process which we developed previous.

4. Future Plans Our research on the model-based diagnosis has thus far lead to the development of the functional model which provides the knowledge base for fault isolation and hypothesization. Some other works are yet to be finished, which we will discuss below:

(1) The detail syntax of the frame-like representation for subsystem is to be completed.

(2) The isolation strategy is to be extended to take care of more complicated causal dependencies. A major challenge will be to detect and handle a "dead-loop" situation.

(3) Representation of physical law associated with the environment-of-response of each subsystem is to be developed.

(4) The interface between fault hypothesization strategy and verification strategy is to be further studied.

5. References

(1) Chien, R.T., Chen, D., and Pan, Y., "Multilevel Semantic Analysis and Problem-Solving in the Flight Domain," Final Report, NASA NCCI-52, July 1981.

(2) Sussman, G., and Stallman, R., "Heuristic Techniques in Computer-Aided Circuit Analysis," IEEE Transactions on Circuits and Systems, May 1975.

(3) Stallman, R., and Sussman, G., "Forward Reasoning and Dependency-Directed Backtracking In a System for Computer-Aided Circuit Analysis," Memo No. 380, Artificial Intelligence Laboratory, M.I.T., Septmber 1976.

(4) Steele, G., and Sussman, G., "Constraints," Memo No. 502, Artificial Intelligence Laboratory, M.I.T., November 1978.

(5) Davis, R. and King, J., "An Overview of Production Systems," Machine Intelligence No. 8.

(6) Davis, R., Buchanan, B., and Shortliffe, E., "Production Rules as a Representation for a Knowledge-Based Consultation Program," Memo AIM-266, Stanford Artificial Intelligence Laboratory, October 1975.

(7) Duda, R.O., Hart, P.E., and Nilsson, N.J., "Development of a Computer-Based Consultant for Mineral Exploration," Annual Report, SRI International, October 1977.

(8) Bennett, J., Creary, L., Englemore, R., and Melosh, R., "SACON: A Knowledge-Based Consultant for Structural Analysis," Memo HPP-78-23, Stanford Heuristic Programming Project, September 1978.

(9) Nii, H.P., and Feigenbaum, E.A., "Rule-based Understanding of Signals," Memo HPP-77-7, Stanford Heuristic Programming Project, April 1977.

(10) van Melle, W., "A Domain-Independent Production-Rule System for Consultation Programs," Proceeding, IJCAI-79.

(11) Davis, R., "Applications of Meta Level Knowledge to the Construction, Maintenance and Use of Large Knowledge Bases," Memo HPP-76-7, Stanford Computer Science Department, July 1976.

(12) Brown, A., "Qualitative Knowledge, Causal Reasoning, and the Localization of Failures," Ph.D. Thesis, AI-TR-362, Artificial Intelligence Laboratory, M.I.T., March 1977.

(13) deKleer, J., "Local Methods for Localizing Faults in Electronic Circuits," Memo No. 394, Artificial Intelligence Laboratory, M.I.T., November 1976.

(14) Feigenbaum, E.A., "The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering," Memo HPP-77-25, Stanford Heuristic Programming Project, August 1977.

(15) Feigenbaum, E.A., Buchanan, B.G., and Lederberg, J., "On Generality and Problem Solving: a Case Study using the DENDRAL Program,"

Machine Intelligence 6, 1971.

(16) Minsky, M., "A Framework for Knowledge Representation," In P. Winston, ed., The Psychology of Computer Vision, McGraw-Hill Book Co., 1975.

Chapter IV

UNDERSTANDING NOVEL MECHANISMS THROUGH INTENTION-DIRECTED RATIONALIZATION

1. Introduction

We are developing the principles and the architecture of a system which understands novel mechanisms through the process of purpose-directed rationalization. Given a novel instantiation, the system which understands the mechanism should be able to generate a consistent explanation of how the instantiation accomplishes its intended behavior, in conceptual vocabulary (jargon), in a framework of the intended operation of the mechanism, and at the appropriate level of detail. At the same time, the explanation must be complete in the sense that it accounts for all the abstract characteristics that define the abstract mechanism. It must also account for all the physical components that appear in the novel instantiation.

This report is divided into two major parts. The first part is intended to put the concept of mechanism understanding in perspective. To this end, we discuss what we are doing and what the theoretical and practical advances are. The second part will focus on the architecture of the mechanism understanding system. We will introduce a mechanism example, and explain each subsystem in the context of its actions on the example. We will focus on the processes and the knowledge sources which define each subsystem.

2. Mechanism Understanding

What is understanding? The dictionary definition of 'understand' is "to grasp or comprehend the meaning intended or expressed by another." In the mechanism domain, the 'meaning intended' is the designed behavior of the physical instantiation, and by a natural extension, the 'another' in the definition is the mechanism's designer.

There are two very important implications here on the content and organization of the knowledge base of the understanding system. Firstly, the system and designer must share a common language (jargon) in order to communicate. This seed knowledge base common to both must include the domain's conceptual vocabulary (which transcends the individual instantiations) as well as the domain's vocabulary of physical component models. Put in the context of the communication model, system and designer must 'talk the same language' in order for the system (as the listener) to understand the instantiation (as the message) produced by the designer (as the speaker).

Secondly, this system knowledge base should be organized into a library of intensional definitions of mechanisms, in order for the rationalization to be an intelligently directed process.

2.1. Intensional Understanding

Consider the following scenario of understanding:

Technician	:	"Look at this schematic diagram."
		"It is supposed to be a DC voltage amplifier."
		"Do you understand it?"
System	:	"yes, I do."

What did the system understand? This scenario is obviously incomplete. Whatever the system understands is useless unless it can be made available in some way (this is in the same vein as 'Write-Only Memory'). In Artificial Intelligence work, the proof of understanding is often expressed as the explanations solicited in response to probing questions. So the question set which the understanding system can deal with represents a good characterization of what it understands. At this juncture, we would like to point out the versatility of the understanding the system is capable of. It can handle question sets which are specific to various applications such as mechanism troubleshooting, mechanism design, and computer-aided mechanism learning. This will be expanded on in a later section, 'Application Advances'.

How did the system understand? This is the second variable which is used to characterize the type of understanding that is achieved. One possible process of understanding is extensional understanding. By extensional understanding, we mean a process which is driven by an extensional definition of the concept in question. The extensional definition of a concept is composed of the set of instantiations of the concept. If complete, the extensional definition is capable of very powerful performance. Put into the context of the mechanism domain, the extensional definition of the 'amplifier' concept would be the set of all amplifier instantiations (and 'novel instantiation' would no longer have meaning). It is clear that an understanding system based on extensional understanding is not elegant and may not be practical.

A second possible process of understanding is intensional understanding. By intensional understanding, we mean a process which is driven by an intensional definition of the concept in question. The intensional definition of a concept is composed of the intrinsic properties which characterizes the abstract concept. It transcends the instantiations and in fact specifies the essential qualities each must satisfy to be an instance. The intensional definition is intrinsically complete. Put into the context of the mechanism domain, the intensional definition of the 'amplifier' concept would be composed of such abstract characteristics as 'gain' and 'DC bias', incorporated in a behavioral description of the definitive operation of amplifiers. Understanding a novel physical instantiation would then be an interpretational process of rationalizing how each abstract characteristic is achieved by the instantiation under study, in the operational context appropriate to that characteristic. Intensional understanding is conceptual, and therefore more intelligent.

The key to understanding by intention-directed rationalization is the existence of intensional definitions in the mechanism domain. We will define the meta intensional definition (describing the types of knowledge that comprise the intensional definition) in a later section on 'Framework Establishment'. It is important to note here that the intensional definition provides global, concept-specific guidance to the understanding process. (To emphasize the key role that intrinsic properties play in understanding by rationalization, we will use the term 'intensional definition'; to emphasize the conceptual domain knowledge organization into intensional definitions of abstract mechanisms, we will use the term 'conceptual definition'; they should be taken to refer to the same definition.) Intensional understanding can be characterized as directed-analysis while extensional understanding can be characterized as table-lookup.

2.2. Explanation Characteristics

Since the proof of understanding is in the explanation, the process of understanding can be viewed as filling in an explanation framework from which answers to directed questions can be drawn. In that perspective, 'depth' of the understanding is manifested as 'goodness' of the explanation. What characterizes a 'good' explanation?

2.2.1. Consistency

An explanation (understanding) must at least be consistent in the sense that it is plausible within constraints imposed by the novel physical instantiation the system is trying to understand. The constraints are those imposed by the behavioral models of the physical components and those imposed by the connective scheme intrinsic to the novel physical instantiation.

2.2.2. Rationalization

In line with the idea of the intensional definition which captures the abstract mechanism, the explanation (understanding) is a rationalization of how the novel physical instantiation does achieve the abstract characteristics which define the abstract mechanism. It explains how the novel physical instantiation conforms to the conceptual definition by bridging the two representations through a causal link. The result is a component level explanation of mechanism level behavior.

2.2.3. In Concept Vocabulary

The explanation (understanding) must incorporate the conceptual vocabulary (jargon) that is the language of the domain. In being able to use the jargon correctly, the system is immediately credited with a high level of domain understanding. Also, explanations should allow the questioner to focus his attention on understanding the content of the explanation of the novel physical instantiation, and not on perhaps unfamiliar terminology. Explanations in domain conceptual vocabulary minimize the language gap. Furthermore, the conceptual understanding is more immediately applicable in expert system applications.

2.2.4. In Operational Context

Since every mechanism is intended to perform some operation, the explanation (understanding) must be housed in the definitive operational context of the abstract mechanism. This includes such contextual knowledge as the intended input signal, the intended phase of operation, the intended output signal, and the intended abstract characteristic highlighted in this phase. The operational context provides the perspective for rationalizing how each abstract characteristic of the intensional definition is achieved by the novel physical instantiation.

2.2.5. At Appropriate Level of Detail

The explanation (understanding) should be organized at various levels of physical detail. This organization corresponds to the basic limitations of the Human Short Term Memory. If there are too many components to keep track of, the human becomes confused. Thus, the concept of the physical substructures (which are mechanisms in their own right) is intrinsic to design and must be accounted for by understanding. The explanation of a mechanism should be in terms of its physical substructures (sub-mechanisms). Each substructure is recursively a mechanism which has its own substructures (sub-mechanisms). The result is a conceptual explanation hierarchy of various levels of detail. In other words, the system should not explain a mechanism of several hundred components all in one breath.

2.2.6. Accounting for Conceptual Definition

The explanation (understanding) should be complete in the sense that it accounts for all parts of the intensional definition of the concept. In other words, the novel physical instantiation must satisfy all the intensional properties of the abstract mechanism.

2.2.7. Accounting for Novel Physical Instantiation

An underlying assumption made by the system is that the mechanism is well-designed. There are no components in the instantiation which do not serve in helping to accomplish some purpose. Correspondingly, the explanation (understanding) should be complete in the sense that it accounts for all components of the novel physical instantiation.

2.3. Theoretical Advances

Now that we have established WHAT we intend to do (Mechanism Understanding), HOW we intend to do it (Intensional Understanding), and HOW we intend to prove it works ('Good' Explanation), we will address the issue of WHY we want to do it. This section covers the theoretical advances of mechanism understanding. The next section covers the advances in expert system applications which are enabled by mechanism understanding.

2.3.1. Intensional Understanding of Novel Physical Instantiations

The understanding of novel instances (be it plans or mechanisms) under the conceptual guidance provided by intensional definitions is a cognitive process which is uniquely human. It forms the basis of his versatility and adaptability. The definition of a system which is capable of understanding novel physical instantiations represents a very key step in understanding the general process of Understanding.

2.3.2. Deeper Knowledge Levels

Surface level knowledge (or the Instantiation Level) is not sufficient to drive a system which understands novel physical instantiations. Accordingly, we have defined a second knowledge level (or the Abstract Mechanism Level) of intensional definitions of mechanism concepts. The intensional definition transcends the physical instantiations of the defined concept. (What comprises an intensional definition is a key Knowledge Organization issue which we will elaborate on in the section on 'Framework Establishment'.)

2.3.3. Conceptual Focus in Abstract Viewpoints

Focus is a key result in Goal-Subgoaling (the decomposition of a problem into several smaller subproblems which may be continued recursively). The issue is how to decompose the problem. We define a viewpoint as one abstract characteristic and the operational context in which to analyze it. By placing the novel physical instantiation into this viewpoint, the intensional definition conceptually directs the system to focus on one intensional property at a time. Thus, the understanding (goal) proceeds, one characteristic (subgoal) at a time, in the appropriate viewpoint.

2.3.4. Component-mechanism Hierarchy

There is a focusing process in the physical plane as well as the conceptual plane. This is manifested as the Component-mechanism Hierarchy in which each physical structure is regarded on the one hand as a mechanism composed of its son nodes, and on the other hand as one component of its father node. The conceptual focus and the physical focus are the basis of explaining the novel physical instantiation in the right context and at the appropriate level of detail.

2.3.5. Conceptual Explanation Hierarchy in Jargon

The generated conceptual explanation is a hierarchical structure which is characterized above in the section on 'Explanation Characteristics'. Explaining in the right context and at the appropriate level of detail is recognized as a key problem in man-machine interfacing (as indicated by its emphasis in the Stanford production systems such as MYCIN [1]).

2.3.6. Skill Knowledge Base

The understood physical instantiations can be organized into a knowledge base we call the Skill Knowledge Base. The Skill Knowledge Base drives the expert system applications, which we will elaborate upon in the following section. From the standpoint of each application, the combination of the understanding system and the Skill Knowledge Base forms a self-extending system. For each novel physical instantiation which the application will act on, the understanding system can extend the Skill Knowledge Base to include it. The issue of knowledge base consistency for the Skill Knowledge Base is really the issue of consistency of the intensional definitions which drive the understanding system. This is desirable since there are fewer intensional definitions and they are relatively well-defined.

2.4. Application Advances

A system which understands novel physical instantiations can support various expert systems in specific applications by providing its understood instantiations as a Skill Knowledge Base. This Skill Knowledge Base acts as the consultant to the application system which is itself probably acting as a consultant [Figure 1]. The medium of exchange is the application-specific question set for which the Skill Knowledge Base will provide solicited answers. Seen in this context, the understanding system can be viewed as a deeper knowledge base into which various application systems can be plugged. We will expand on three potential applications on which an understanding system has great impact. They are by no means an exhaustive applications list.

2.4.1. Computer-aided Learning

An immediately appropriate application of an understanding system is computer-aided learning. This is in contrast to computer-aided instruction in which pre-programmed, static lessons are projected on the CRT screen in fixed order. There is very little student input simply

EXPERT TROUBLESHOOTING SYSTEM

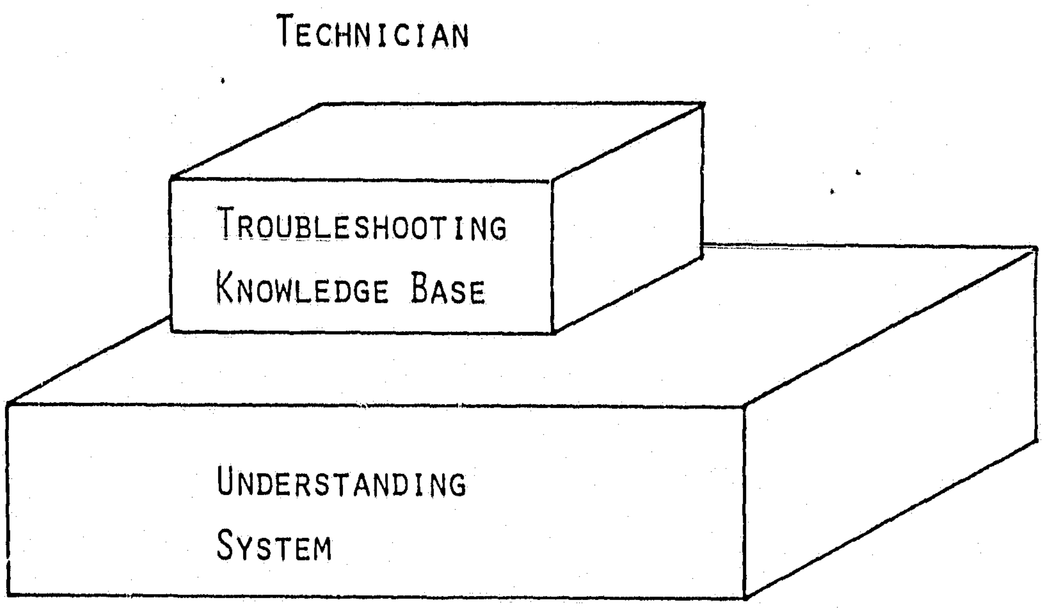


FIGURE 1

because the program is not intelligent enough to handle any deviations from its planned lessons. New lessons must be tediously programmed by hand. In effect the computer-aided instruction system learns by being programmed. It certainly cannot handle novel designs which the student might have seen in the textbook, but which do not appear as a selected example in the pre-programmed lessons.

With the system which understands novel physical instantiations, the student has a dynamic system which can understand the novel design and explain it to him in conceptual vocabulary. The explanation is consistent, complete, and at an appropriate level of detail from the substructure level to the primitive component level. In this way, it caters to various levels of students automatically.

The computer-aided learning system is self-extending since each novel physical instantiation represents another lesson added to the Skill Knowledge Base. It can draw upon this knowledge base in response to new student requests to explore abstract mechanisms further. In effect, the student who first raises a novel physical instantiation has taught the understanding system which in turn teaches other students. The computer-aided learning system learns by rationalization. It automatically learns to teach automatically.

2.4.2. Computer-aided Design

With a Skill Knowledge Base (design library) to draw from, the computer-aided design system can propose a basic design in response to performance specifications desired by the designer. Furthermore, with a complete intensional definition, the system is able to intelligently elicit design decisions, in conceptual designer language, which the designer might have overlooked. For example, once a designer specifies that he wants an amplifier, the system might inform him that he must specify whether it is to be a voltage amplifier or power amplifier. Since the Skill Knowledge Base includes the causal bridge between the abstract definition and the physical instantiation, the performance specifications can be causally backtraced to what component parameter values should be changed, and to what new values. In this way, the computer-aided design system can act as an apprentice designer.

In another context, the computer-aided design system can act as monitor. Novel physical designs can be submitted to the system which tries to understand it under the direction of the intensional definition which any design must satisfy. Design errors can be caught if a consistent explanation of an abstract characteristic cannot be reached. The inconsistent explanation can be offered as partial information to help clear up the error, rather than just stating that something is wrong. Design oversights can also be caught since the intensional definition serves as a conceptual checklist of intrinsic design considerations. Again, as above, novel designs which pass the tests are incorporated into the Skill Knowledge Base, perhaps to be suggested as a basic design later on down the line.

2.4.3. Computer-aided Troubleshooting

Intelligent troubleshooting must proceed from an understanding of the mechanism under study. This understanding comes in two parts: the intensional definition of the abstract mechanism, and the rationalized novel physical instantiation. Each part plays a role in intelligent troubleshooting.

The intensional definition contains knowledge of the abstract characteristics which define the mechanism, and the intended operational context of the mechanism. These serve to define the functional test procedure, which appropriately should be synthesized at the abstract mechanism level. What the intensional definition provides the troubleshooting system is knowledge of WHAT to look for in the output (abstract characteristic) and the CONTEXT in which to look (operational context). Since there are multiple viewpoints which decompose the abstract mechanism into multiple abstract characteristics, there is correspondingly one functional test specified per viewpoint. By combining these functional tests into the test procedure and executing the test procedure, some of the abstract characteristics will be discovered to be in error while others are still as they should be. Thus, the intensional definition drives the troubleshooting system to identify the fault signature in terms of the abstract characteristics which define the abstract mechanism. This corresponds to the test procedure followed by the human troubleshooter.

The rationalized novel physical instantiation contains knowledge of how each abstract characteristic is achieved by a substructure of the instantiation under study. From these links of physical substructures to abstract characteristics, and the two sets of 'good' and 'bad' abstract characteristics determined by functional testing, the identity of the component causing the fault can be localized in the following way. Initially the troubleshooting system assumes that every component is in the candidate set of faulty components. For every abstract characteristic that was determined to be 'good' in the functional testing phase, the system assumes that the corresponding physical substructure is 'good'. This is certainly a heuristic but a very reasonable one, although it is possible that two components in a 'good' structure may be faulted in a complementary manner to mask each other. If we make the single fault assumption, then it is no longer a heuristic, but rather is always true. By applying this heuristic, the components in all the 'good' substructures are removed from the candidate set of faulty components. A second heuristic can be applied at this point - the faulty behavior should be explainable be the smallest set of bad components possible. In most cases, there should be only one faulty component. It therefore seems reasonable to order the candidates remaining in the candidate set by the number of 'bad' abstract characteristics in which they play a role (appear in the corresponding physical substructure). Those components which appear in every 'bad' physical substructure should certainly be checked first. If we make the single fault assumption, then only those candidate components which appear in every 'bad' physical substructure are kept in the candidate set. All other components are inferred to be good.

In the case of parameter drift faults (the component parameter drifts high or drifts low), there is yet another type of information provided by the rationalized novel physical instantiation. The abstract characteristics are related to a corresponding physical substructure. But the system also has the equational relationship between the abstract characteristic and the parameters of the components in that physical substructure. By hypothesizing a particular candidate component (in the partial order determined above), the troubleshooting system can determine how that component must have faulted (direction of parameter drift) to cause the faulty behavior, by using the equational relationship. If a parameter of a candidate component is determined to have drifted high in one substructure (to explain one faulty abstract characteristic) but low in another, it is reasonable to question whether the component is the culprit. Again this is a heuristic since more than one component may be faulty. If we make the single fault assumption, then that candidate described above is inferred to be good.

Troubleshooting is a very difficult application which has recently attracted growing interest (all these comments are applicable to computer-aided instruction and computer-aided design). We do not presume to denigrate its difficulty. Building the knowledge base of troubleshooting techniques is undoubtedly a complex problem in both knowledge organization and knowledge representation. However, we do claim that an understanding system would play a key role in facilitating the concept of functional testing, which has recently been the focus of state of the art research [2, 3]. We have presented a first cut indication of how an understanding system would play that role.

3. Focusing on the System

3.1. Mechanism Understanding System

The input to the mechanism understanding system is composed of two parts. The primary input is a description of the novel physical instantiation containing such information as component names, component types, and the connection schemes which define the physical structure. The secondary input is the mechanism name which identifies the intended purpose of the novel physical instantiation. The output is proof of conceptual understanding of the novel physical instantiation. That proof is manifested as a hierarchical explanation of how the physical structure achieves the abstract characteristics which make up the intensional definition referenced be the mechanism name. The hierarchical explanation is in conceptual vocabulary, housed in the intended operational context, and complete in accounting for the intensional definition and the novel physical instantiation [Figure 2].

The same mechanism name may apply to several different novel physical instantiations. For example, there are various physical instantiations of the DC voltage amplifier [Figure 3]. This is the power of intensional understanding. The abstract mechanism of DC voltage amplifier transcends its various physical instantiations, including some that may have not yet been designed. The understanding system knows what intrinsic properties any design, old or new, must satisfy to legitimately be called a DC voltage amplifier. So the understanding system knows what to look for, and in what operational context, in rationalizing whether a novel physical instantiation can legitimately be called a DC voltage amplifier or not.

The understanding system is composed of four processes [Figure 4]: Framework Establishment, Physical Conceptualization, Behavior Verification, and Experience Incorporation.

Framework Establishment is the process of placing the novel physical instantiation into various viewpoints (one abstract characteristic and the operational context in which to analyze it). What is happening

MECHANISM UNDERSTANDING

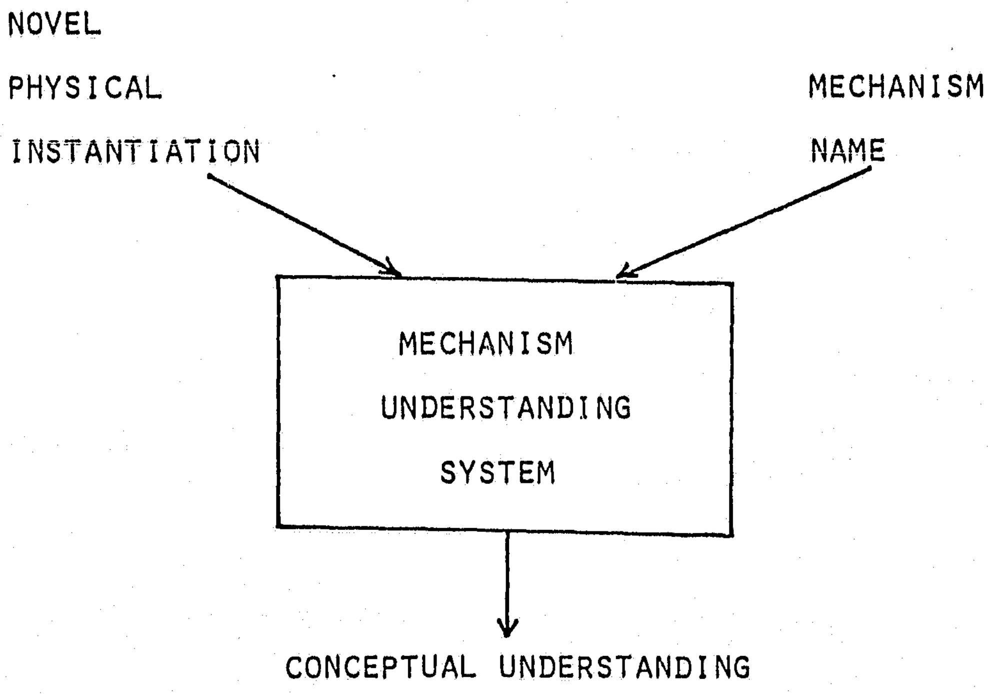
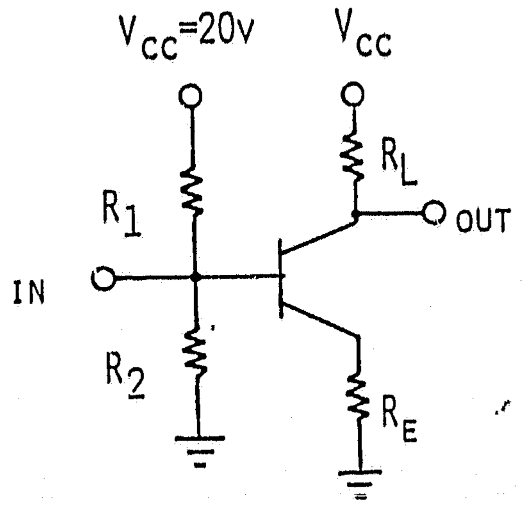
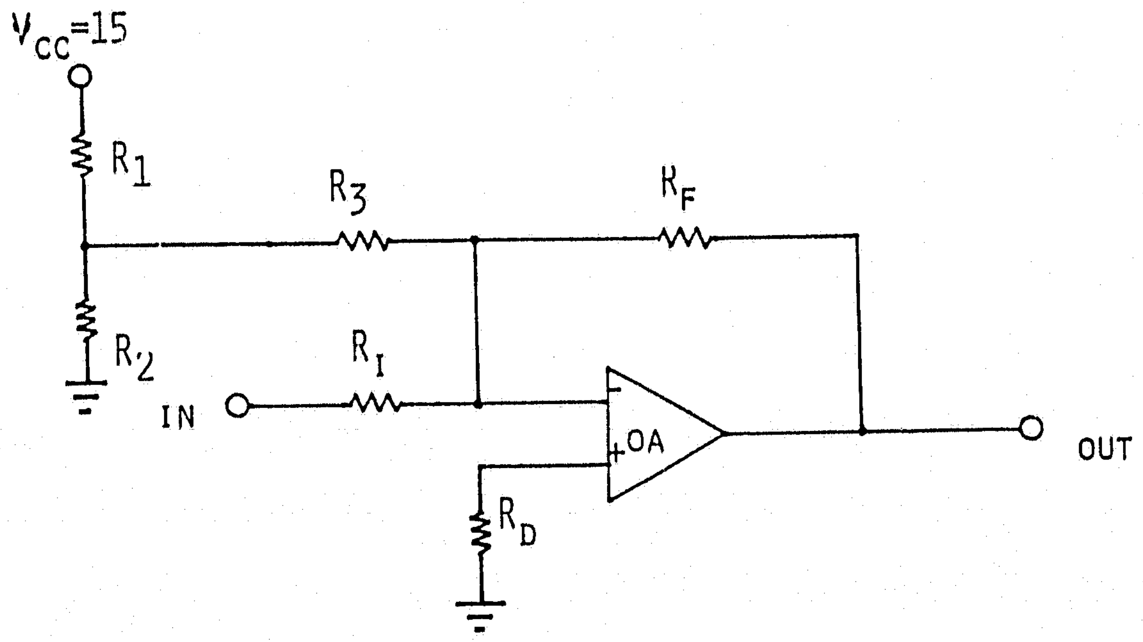


FIGURE 2

MECHANISM UNDERSTANDING - INPUTS





DC VOLTAGE AMPLIFIER

FIGURE 3

UNDERSTANDING - PROCESS EXPANSION

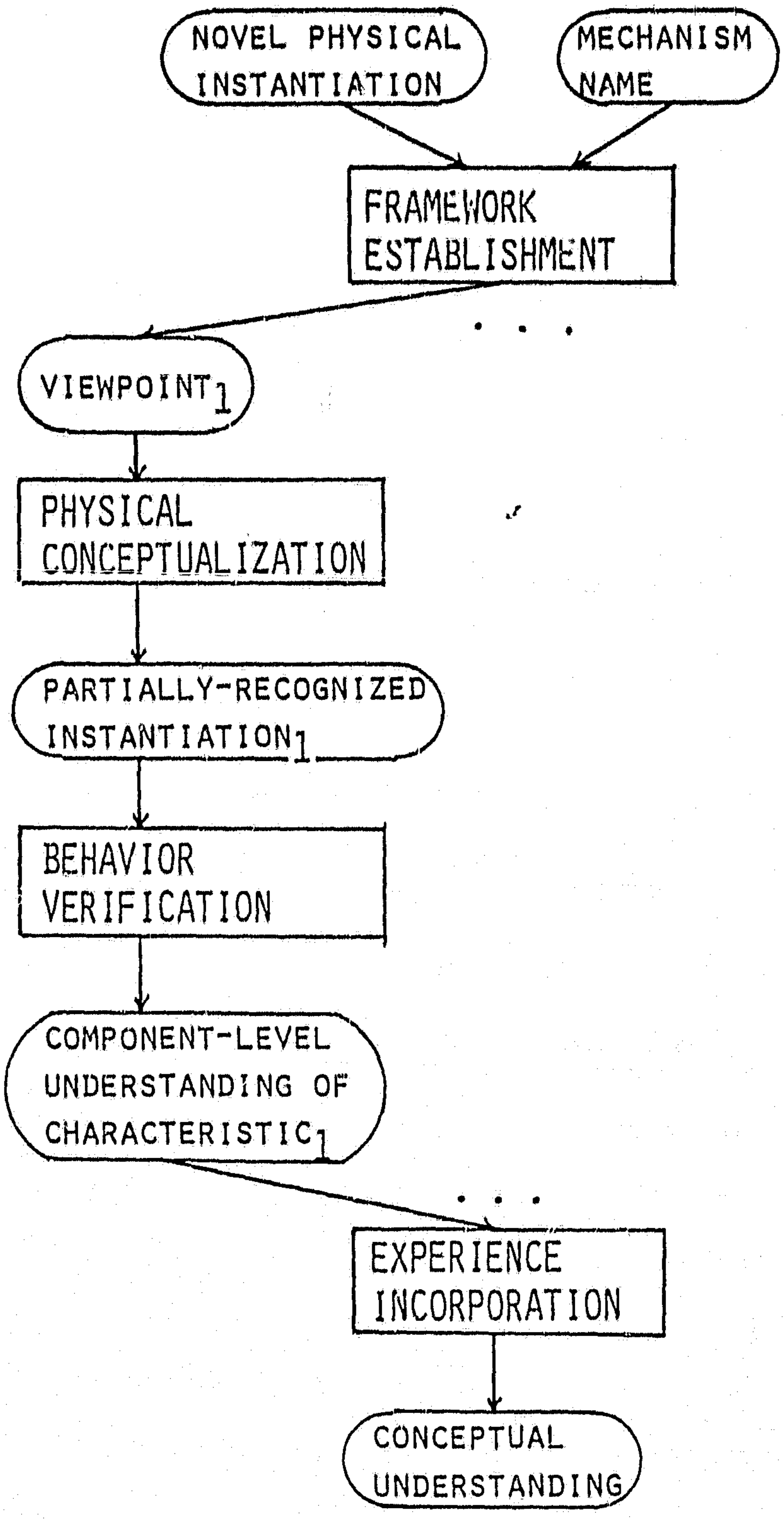

NOVEL PHYSICAL INSTANTIATION
MECHANISM NAME
FRAMEWORK ESTABLISHMENT
. . .
VIEWPOINT$_1$
PHYSICAL CONCEPTUALIZATION
PARTIALLY-RECOGNIZED INSTANTIATION$_1$
BEHAVIOR VERIFICATION
COMPONENT-LEVEL UNDERSTANDING OF CHARACTERISTIC$_1$
. . .
EXPERIENCE INCORPORATION
CONCEPTUAL UNDERSTANDING

FIGURE 4

is that the intensional definition conceptually guides the understanding system to focus on the novel physical instantiation, one intrinsic property at a time.

Physical Conceptualization is the process of moving the structural description as far up the Component-mechanism Hierarchy (see section on Theoretical Advances) as allowed by experience. The experience is the collection of understood physical instantiations which form the Skill Knowledge Base. Here again, the attention of the understanding system is being focused, this time conceptually guided by experience gained through past encounters with other novel physical instantiations. The result is still a structural description, but with fewer components in a simpler connection scheme.

Behavior Verification takes this simpler structural description, in its various viewpoints, and generates the component-to-mechanism link. That link explains how the physical structure achieves the abstract characteristic focused on, in the appropriate context, in that viewpoint.

Experience Incorporation is the process of inserting the understood novel physical instantiation into the Skill Knowledge Base. One primary task is to make the experience gained in this session available for application in the Physical Conceptualization process in future sessions. Another is to coordinate the hierarchical explanation of the novel physical instantiation and make it available stand-alone, or in the context of one of the various applications an understanding system can support.

In the next four sections, we will delve into the four processes that comprise the understanding system. We will focus on the knowledge sources which drive them and show the key snapshots of the data base as it passes through the understanding system. To complement the explanation of the workings of the various processes, one example will be rationalized. To serve that purpose, we will use the transistor instantiation of the DC voltage amplifier [Figure 3].

3.2. Framework Establishment

The key knowledge source of the Framework Establishment process is the conceptual definition which is intensional in nature. The purpose of Framework Establishment is to place the novel physical instantiation into various viewpoints as dictated by the conceptual definition corresponding to the mechanism name provided as input. In this way, the conceptual definition breaks the problem of understanding down into various subproblems of understanding how a particular abstract characteristic is achieved by the novel physical instantiation. The result is conceptually-directed focus of analysis, one viewpoint at a time, by the understanding system.

The meta conceptual definition (describing the types of knowledge that comprise the conceptual definition) has two basic types of conceptual knowledge [Figure 5]. The first is a list of abstract characteristics. The second is a state transition diagram capturing abstract mechanism behavior. To explain the contents of the conceptual definition, we will use the one corresponding to the DC voltage amplifier.

CONCEPTUAL DEFINITION

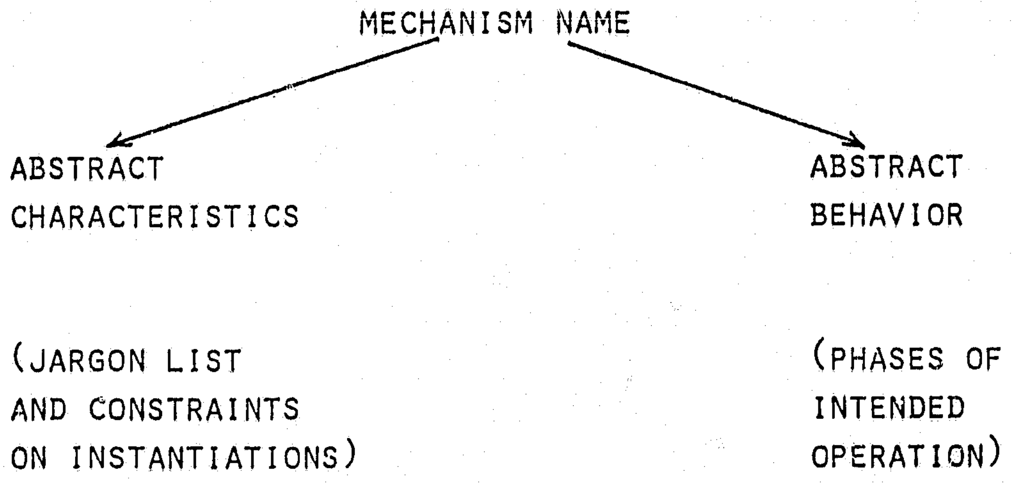

MECHANISM NAME
ABSTRACT CHARACTERISTICS
ABSTRACT BEHAVIOR
(JARGON LIST AND CONSTRAINTS ON INSTANTIATIONS)
(PHASES OF INTENDED OPERATION)

FIGURE 5

The list of abstract characteristics, a list of jargon specific to the amplifier [Figure 6], is a vocabulary list of intrinsic properties which transcend any physical instantiation. Such conceptual vocabulary as 'bias' and 'gain' are performance characteristics which describe definitive behavior. Such conceptual vocabulary as 'class of operation', 'signal type', and 'frequency range' are classificational characteristics which partition the set of amplifiers in the pragmatic taxonomy intrinsic to the domain. Associated with each abstract characteristic is a constraint description. For example, the bias must be a DC value and the gain must be a numerical constant. The class of operation can be labelled in four possible ways (A, B, AB, or C) each of which is well-defined. The signal type can be labelled in two possible ways, and so on. This list represents the static vocabulary used by those initiated into the domain. There is not yet any direct knowledge of operation.

The state transition diagram which is intended to capture the abstract mechanism behavior [Figure 7], is a directed graph with two types of nodes, state nodes (indicated by 'S:') and action nodes (indicated by 'A:'). Each state node represents a viewpoint in which one abstract characteristic of the conceptual definition should be determined. The action-state sequence leading up to the state in question represents the establishment of the proper operational context in which to do the analysis. In the 'bias' state, the system is directed to analyze how the novel physical instantiation achieves the bias. The operational context indicates that the power is on, but that there is no input signal. This state transition diagram is not intended to define

ABSTRACT CHARACTERISTICS - AMPLIFIER

BIAS	DC
GAIN	CONSTANT
CLASS OF OPERATION	A,B,AB,C
SIGNAL TYPE	V,P
FREQUENCY RANGE	DC, AUDIO, VIDEO
	...

FIGURE 6

ABSTRACT BEHAVIOR - AMPLIFIER

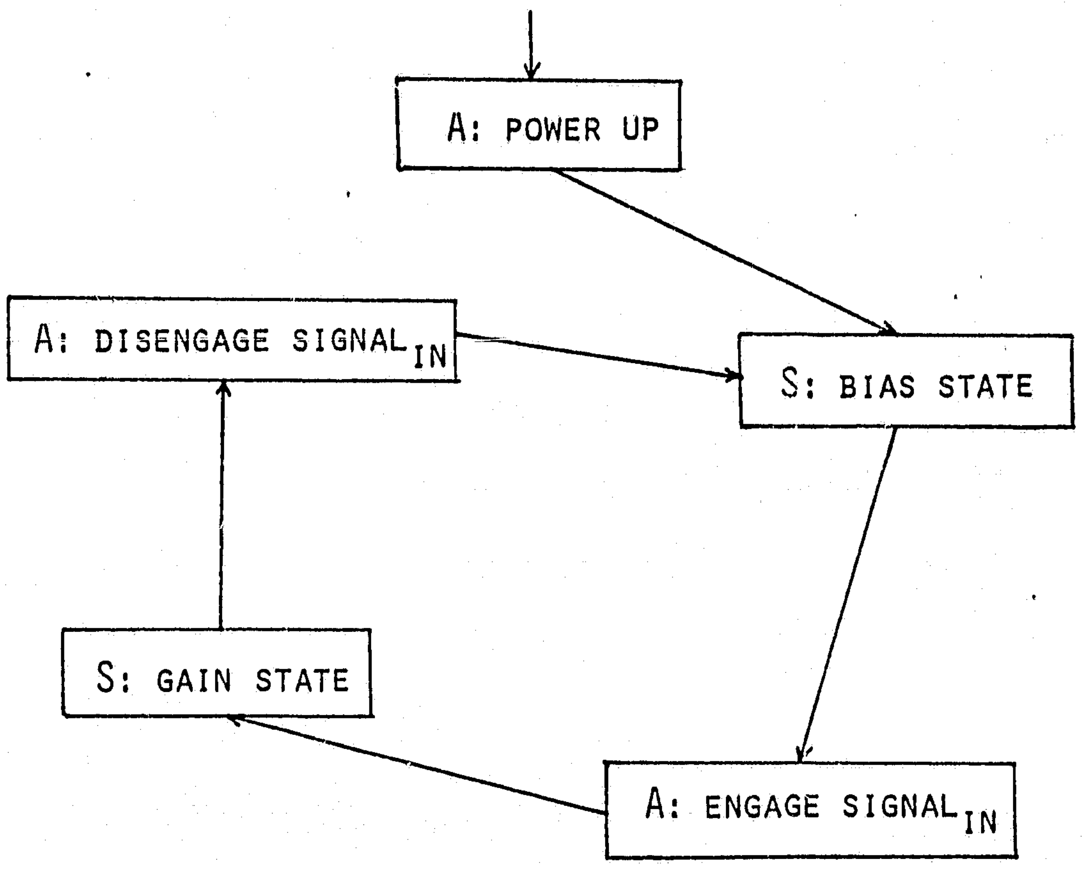


FIGURE 7

the only way in which the mechanism may operate. Rather, it is one way which puts the mechanism through its paces thoroughly. It serves as an anchor for directed analysis. The state transition diagram represents the intensional knowledge of the dynamic (procedural) aspects of the abstract mechanism.

The conceptual definition provides the system with knowledge of what to look for and in what context to look. It does so through a series of viewpoints. The meta viewpoint [Figure 8] holds four basic types of knowledge. The abstract characteristic tells the understanding system what to focus on in this viewpoint. The context contains the history (action-state sequence leading up to the state corresponding to the current viewpoint), the proper input signal, and the expected output signal. The focused physical structure ignores physical components which are not relevant to this viewpoint. The component-level rationalization is a placeholder for the component-level explanation of how the novel physical instantiation achieves the abstract characteristic. The focused physical structure will be tailored to reflect this explanation. The viewpoints corresponding to 'bias' and 'gain' for the DC voltage amplifier are shown in Figure 9.

The output of the Framework Establishment process is a set of viewpoints, which are passed to the Physical Conceptualization process. Framework Establishment has conceptually decomposed the understanding problem by defining the intensional set of understanding subproblems for the rest of the understanding system to focus on.

Viewpoints - Structure

ABSTRACT CHARACTERISTIC

CONTEXT

- HISTORY
- INPUT SIGNAL
- OUTPUT SIGNAL

FOCUSED PHYSICAL STRUCTURE

COMPONENT-LEVEL RATIONALIZATION

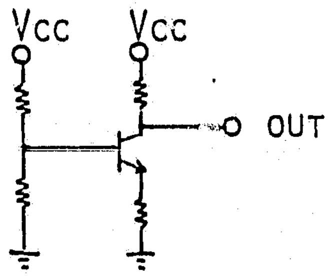
Figure 8

VIEWPOINTS - DC VOLTAGE AMPLIFIER

BIAS VIEWPOINT

ABSTRACT CHARACTERISTIC:	BIAS	DC
CONTEXT:	POWER ON	
	NO SIGNAL IN	

PHYSICAL STRUCTURE:



GAIN VIEWPOINT

ABSTRACT CHARACTERISTIC:	GAIN	CONSTANT
CONTEXT:	POWER ON	
	SIGNAL IN = V_{DC}	

PHYSICAL STRUCTURE:

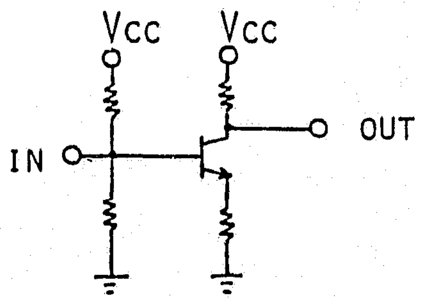


FIGURE 9

3.3. Physical Conceptualization

The key knowledge source of the Physical Conceptualization process is the Semantic Template Hierarchy which represents the accumulation of experience from past sessions with other novel physical instantiations. It resides in the Skill Knowledge Base. The purpose of Physical Conceptualization is to move the structural description of the novel physical instantiation as far up the Component-mechanism Hierarchy as allowed by the Semantic Template Hierarchy. In this way, the Semantic Template Hierarchy simplifies the problem of understanding by simplifying the structural description of the novel physical instantiation. The result is conceptually-directed focus of analysis, at the highest level of structural description possible, by the understanding system.

The meta Semantic Template has three types of knowledge [Figure 10]. The first is a structural pattern of several physical components connected in a predefined connection scheme. The second is a list of semantic constraints which the structural pattern must satisfy. The third is a behavioral description of the structural pattern considered as a single, new component (actually a pointer to the conceptual definition for which this semantic template is one physical instantiation). The Semantic Template can be regarded as a transformational operator which looks to perform syntactic matching and semantic matching against the novel physical instantiation. If both types of match constraints are satisfied, the Semantic Template transforms the pattern in the novel physical instantiation into the one single new component. The result is a new level in the Component-mechanism Hierarchy corresponding to this instantiation. The base level of the hierarchy is the initial struc-

SEMANTIC TEMPLATE

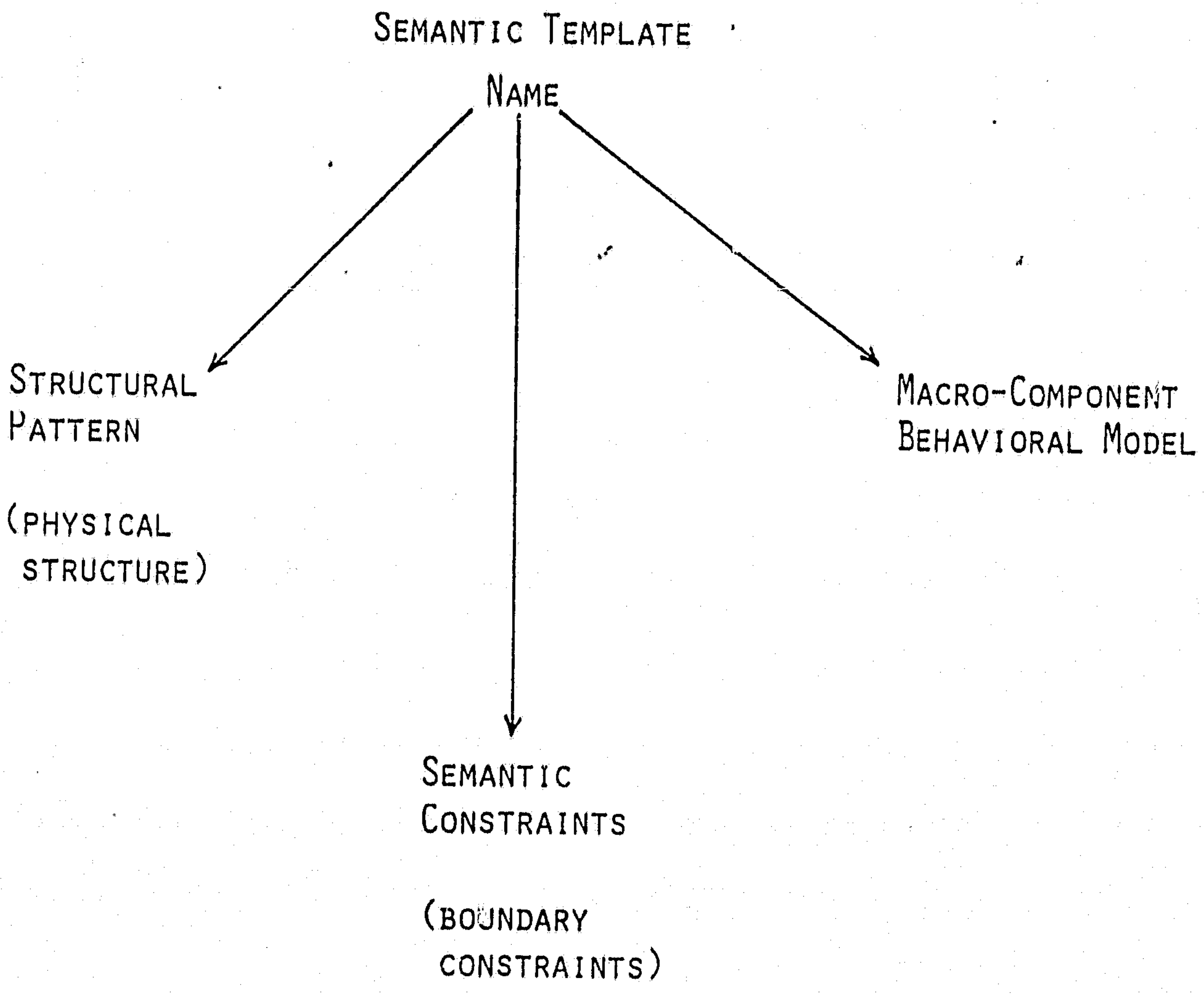


FIGURE 10

tural description of the novel physical instantiation. Note that it doesn't matter how many primitive components comprise the structural pattern, once it is transformed by the Semantic Template into a single new component. This is the power of the Component-mechanism H Hierarchy. The new component has its abstract characteristics just as did each primitive component. The Semantic Template transformation simplifies the syntactic description without losing any semantic knowledge.

The structural pattern is the basis of the syntactic matching. This process is classical in the field of pattern matching and will not be discussed here.

The list of semantic constraints which the structural pattern must satisfy is the basis of semantic matching. The power of coordinating knowledge-based semantic matching with syntactic pattern matching also comes from the seed knowledge base of intensional definitions which is the heart of the understanding system. Each leaf Semantic Template is associated with the abstract mechanism for which it is one possible physical instantiation. Each new leaf Semantic Template therefore represents at least one previous session through the understanding system. The semantic constraints are generated from these previous interactions by an induction process which will be explained in greater detail in the section on Experience Incorporation.

One type of semantic constraints is the parameter relationships among the components in the structural pattern. For example, in the operational amplifier instantiation of the DC voltage amplifier [Figure 3], the drift resistor 'Rd' should have the same impedance value as that seen by the inverting input of the operational amplifier in order for

'Rd' to be acting as a drift resistor. Another type of semantic constraint is the voltage-current boundary conditions which must be met for the structural pattern to behave as intended. For example, in the transistor instantiation of the DC voltage amplifier [Figure 3], the tap current of the voltage divider (R1 and R2) must be approximately zero for the physical structure to be acting as a voltage divider. The class of semantic constraints represents the physical context (of neighboring structures) that the structure in question must have in order to properly operate. The Semantic Template corresponding to the voltage divider which appears in our vehicle example is shown in Figure 11 and the result of matching in the bias viewpoint of the DC voltage amplifier is shown in Figure 12. The result of matching is structurally simpler but conceptually still the same.

Now that we know that the system will use Semantic Templates, the obvious question is how does the system know where on the novel physical instantiation to begin matching? It knows where because it uses the concept of anchor points, and the process of anchor point propagation. Anchor points represent physical boundary points where meaningful structures must begin and end. Clearly, the initial set of anchor points contains the input node, output node, Vcc node, and GND node. Semantic Template matching begins at anchor points. As structures are matched and transformed by Semantic Templates, the new boundary points identified by the match are entered into the set of anchor points, and thus anchor point propagation. In the DC voltage amplifier [Figure 12], the matching of the voltage divider identifies the transistor base node as a new anchor point.

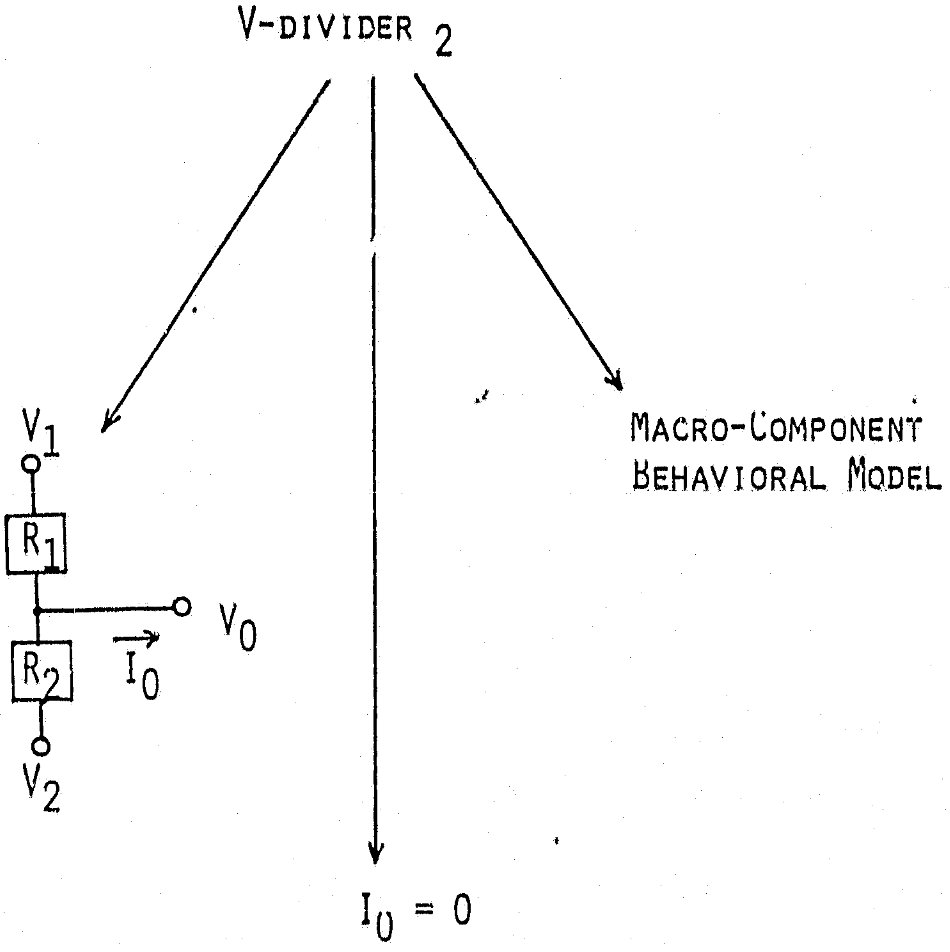
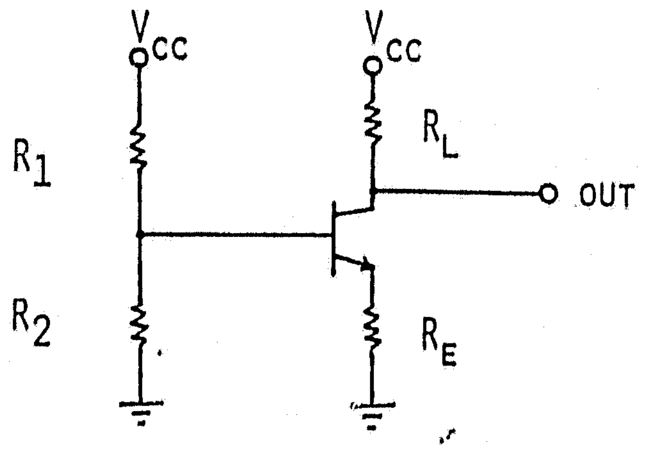

V-DIVIDER 2
V_1
R_1
V_0
R_2
$\overrightarrow{I_0}$
V_2
MACRO-COMPONENT
BEHAVIORAL MODEL
$I_0 = 0$

FIGURE 11

ORIGINAL PAGE IS
OF POOR QUALITY

BIAS VIEWPOINT - DC VOLTAGE AMPLIFIER



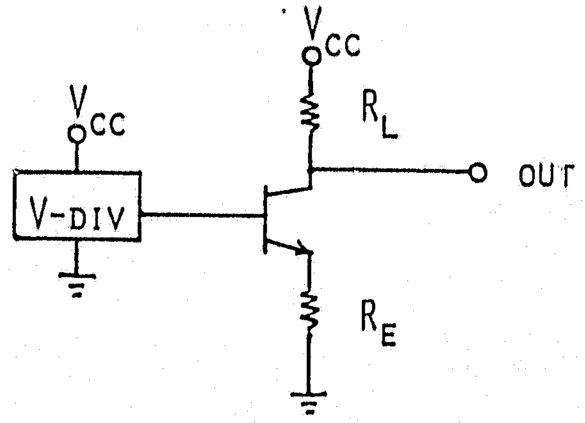
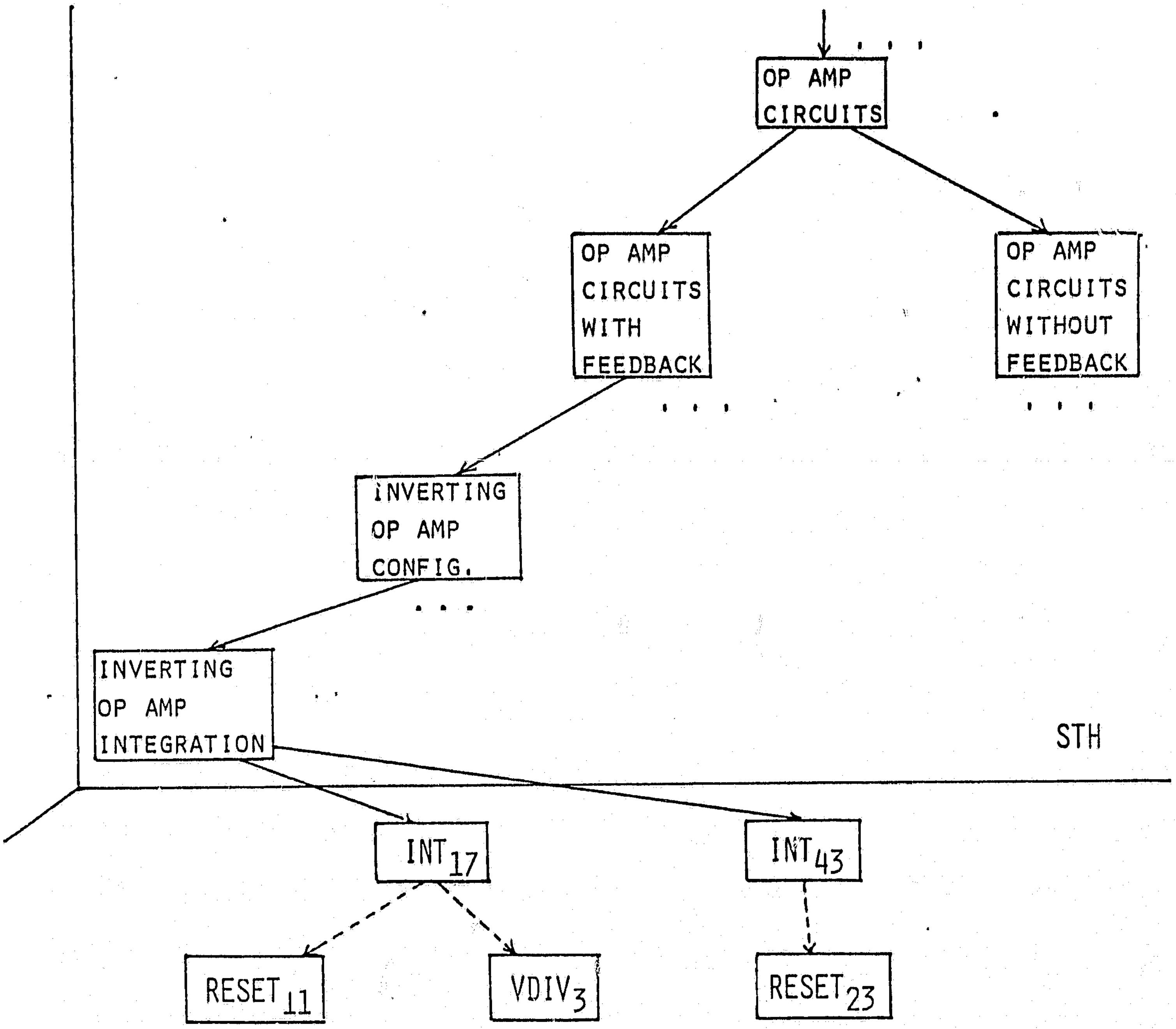


FIGURE 12

Once we know where on the novel physical instantiation the system begins Semantic Template matching, the next question is how does the system know which Semantic Templates are most likely to succeed in matching and should therefore be tried first? Some heuristic guidance is provided by the Semantic Template Hierarchy [Figure 13]. The organization of this structure is based on the pragmatic domain tendency to classify mechanisms physically by key components. In the circuit domain, these are such active elements as transistors and operational amplifiers (we speak of the 'family of operational amplifier circuits' and the 'family of gas-engine-powered vehicles'). Thus the organization of the Semantic Template Hierarchy is a set of physical classification trees that comprehensively partition circuit families. Each succeeding level of the hierarchy represents physical specialization (the hierarchy is a generalization-specialization tree). Thus if the operational amplifier is a component in the novel physical instantiation, the understanding system begins traversal down the operational amplifier circuit family tree which can be viewed as a decision tree. We do not claim that this is the only possible organization, but only that it has credence as a common, efficient pragmatic organization.

The output of the Physical Conceptualization process are simplified physical structures each within its proper viewpoint, which are passed to the Behavior Verification process. Physical Conceptualization has, under conceptual guidance from past experience, simplified each understanding subproblem for the rest of the understanding system to focus on.

SEMANTIC TEMPLATE HIERARCHY


OP AMP CIRCUITS
OP AMP CIRCUITS WITH FEEDBACK
OP AMP CIRCUITS WITHOUT FEEDBACK
INVERTING OP AMP CONFIG.
INVERTING OP AMP INTEGRATION
STH
INT17
INT43
RESET11
VDIV3
RESET23
CMH

C-2

FIGURE 13

3.4. Behavior Verification

The key knowledge source of the Behavior Verification process is the constraint models of components and connections which must be satisfied in order for the behavior to be consistent. Ambiguities arise in that the constraint models may allow more than one consistent explanation of possible behavior. This occurs because the novel physical instantiation is a context-free mechanism unless knowledge of designer intentions are somehow made known to the understanding system. In the scenario of understanding presented by intention-directed rationalization, this knowledge is made available in the intensional definitions which comprise a seed knowledge base of the understanding system. Thus, Behavior Verification is conceptually guided to work within viewpoints defined by Framework Establishment. The purpose of Behavior Verification is to automatically generate a component level explanation of how the novel physical instantiation achieves the abstract characteristic specific to the viewpoint, in the operational context specific to the viewpoint. One way in which it can do this is the process of constraint propagation [4]. By doing so, the understanding system creates a causal link between the abstract mechanism and the components of the novel physical instantiation. The result is an equational relationship between the abstract mechanism characteristic and abstract component characteristics of its corresponding substructure.

The component constraint model is generated deterministically from the component behavior model [Figure 14], which is the corresponding intensional definition of the component as an abstract mechanism (primitive components are leaf nodes in the Component-mechanism Hierarchy).

COMPONENT BEHAVIORAL MODEL

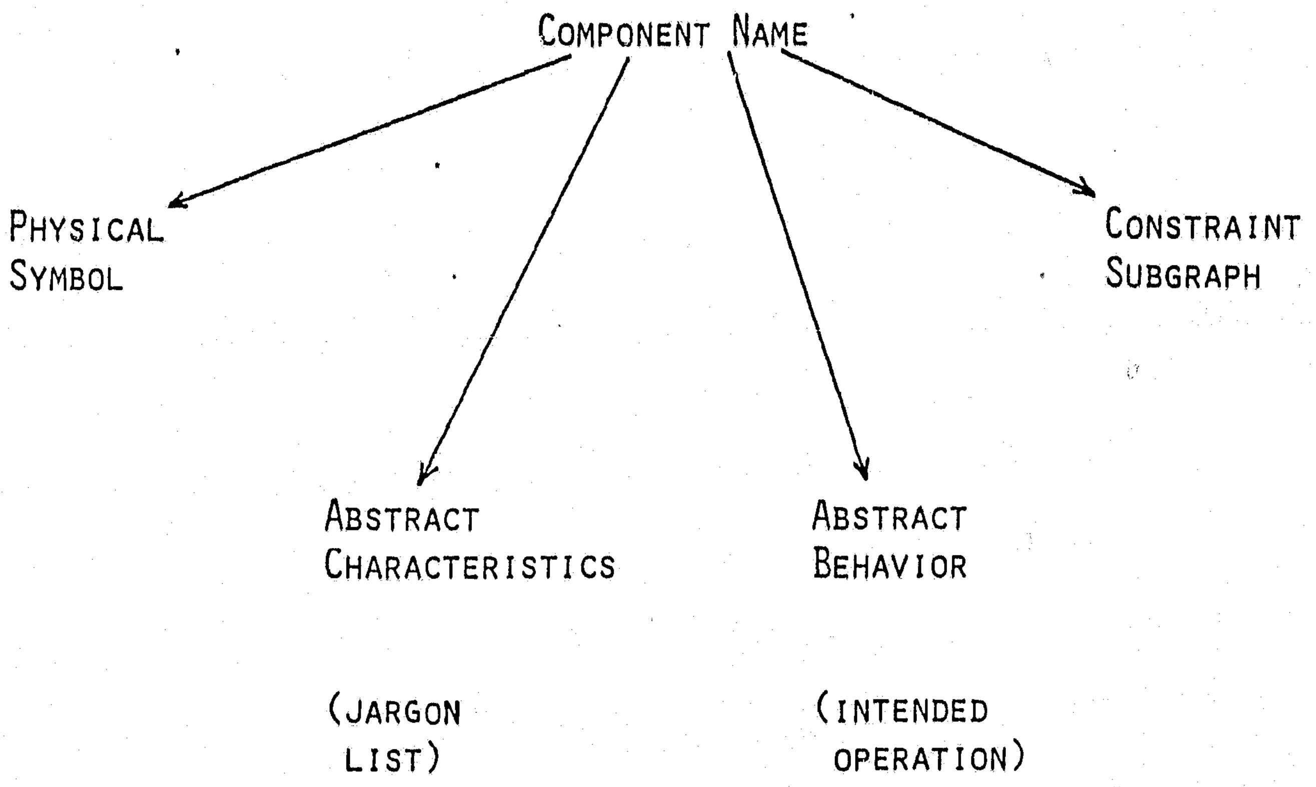


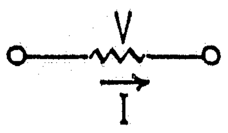
FIGURE 14

In other words, each component in the physical structure passed from the Physical Conceptualization process has a corresponding constraint model based on its abstract behavior. For example, the resistor is a primitive component which behaves as specified by Ohm's Law [Figure 15]. Its constraint model or subgraph is composed of the three variables in the equation and the three corresponding demons [5] which monitor the data base. Simply stated, the demon is activated when all but one of the variables are instantiated (take on values) in the data base. It uses the behavioral description to determine the value of the last variable which it then enters into the data base, hopefully triggering other demons. The voltage divider, while not a primitive component, is still a component. It correspondingly has a behavioral description in terms of key variables just as did the resistor. It therefore also has a constraint subgraph [Figure 15].

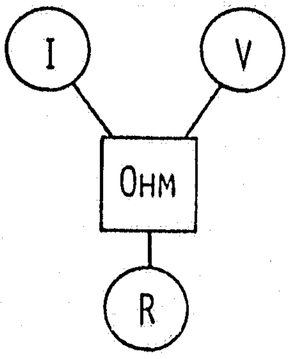
The constraint network is the connection of the component constraint subgraphs according to the connection scheme intrinsic to the novel physical instantiation. It is generated deterministically from the physical structure in each viewpoint as focused by the Framework Establishment and Physical Conceptualization processes. One type of connecting 'glue' is the connection constraint specified by Kirchoff's conservation laws. For example, for the bias viewpoint of the DC voltage amplifier [Figure 16], the current coming out of the emitter resistor, Re, must be the current going into the emitter terminal of the transistor, Q, as specified by KCL. Another type of connecting 'glue' is the connection of component terminals to a common node. For example, the output terminal of the voltage divider, V-div, shares the same node

CONSTRAINT SUBGRAPHS

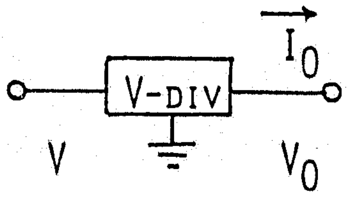
R



$V = I \times R$



V-DIVIDER



$V_0 = F \times V$
$I_0 = 0$

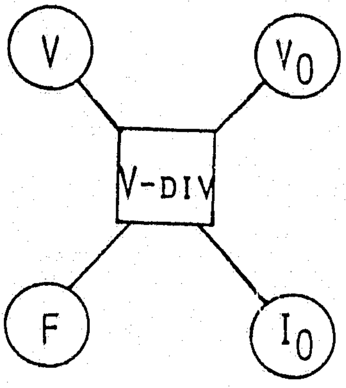
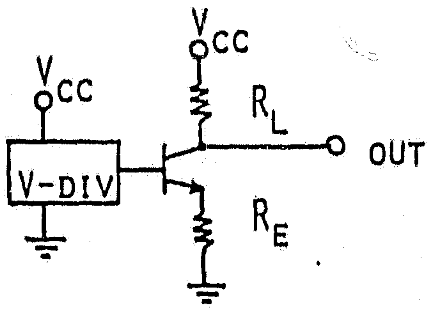


FIGURE 15

CONSTRAINT NETWORK - BIAS VIEWPOINT



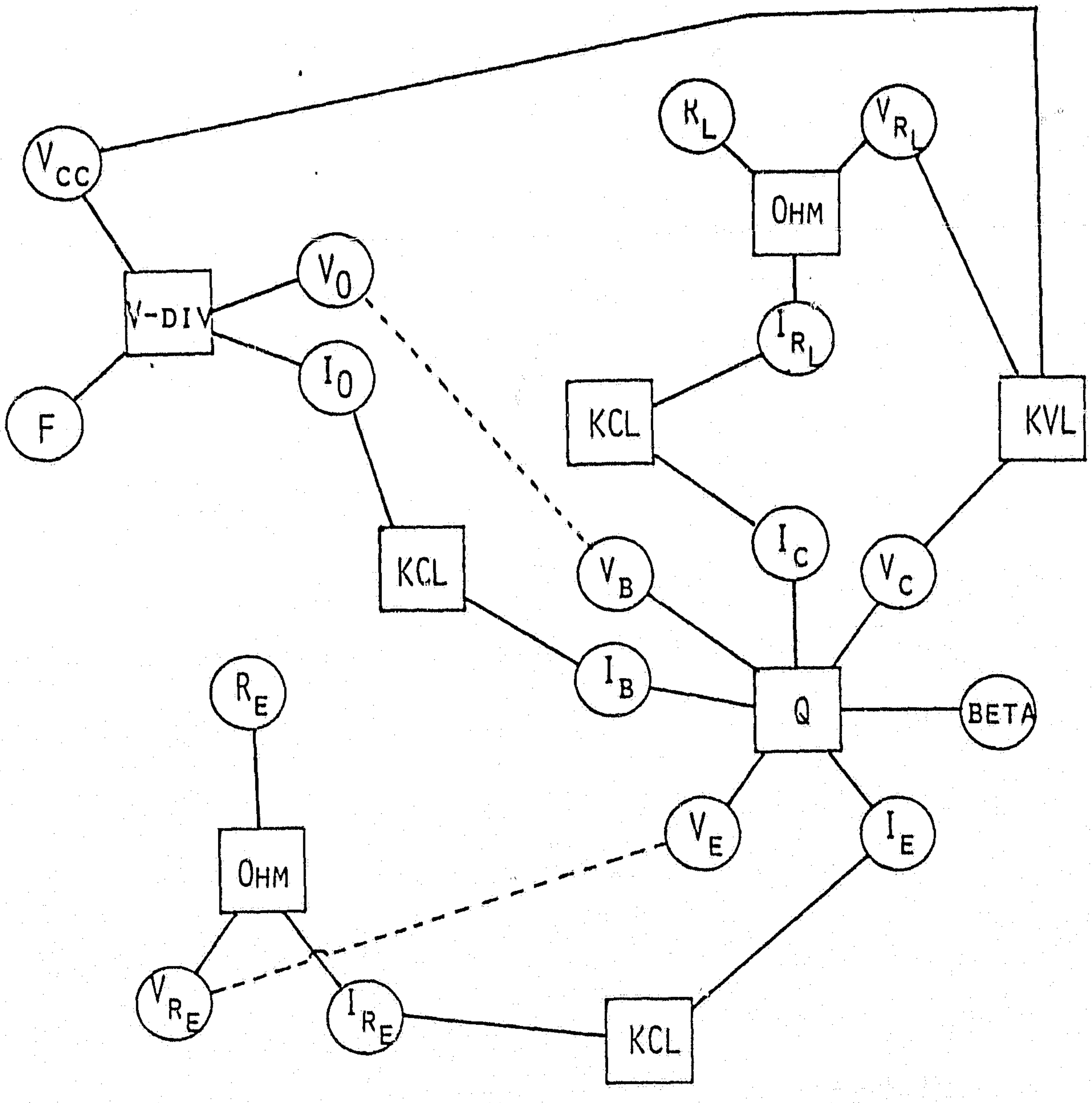


FIGURE 16

as the base terminal of the transistor, Q, so the output voltage of the voltage divider is identically the base voltage of the transistor. Any explanation which is generated from this constraint network must be consistent.

The constraint propagation process begins with the insertion of known variables into the data base, which is being monitored by the demons associated with each component whose subgraph is a part of the constraint network. In the case of the DC voltage amplifier, the explanation which is generated is an equational derivation [Figure 17] which can be viewed as a mathematical proof from hypothesis ('given the physical structure and the context ...') to conclusion ('then the output is indeed the abstract characteristic in question ...'). The result is an equational relationship between the abstract mechanism characteristic and the abstract component characteristics of the pertinent physical structure.

It is appropriate here to discuss the challenging and inquisitive nature inherent in an understanding system. Because of the accountability requirement for explaining all of the intensional properties of the conceptual definition, the understanding system is able to challenge what it perceives to be an incomplete novel physical instantiation, if it can verify that an abstract characteristic is not achieved. For example, in explaining the bias of an amplifier, the equational relationship should satisfy the intensional constraint that the bias is a DC value. If not, the understanding system knows that the novel physical instantiation should not be called an amplifier and can back up its challenge. Because of the accountability requirement for incorporating

Explanation - How Bias Achieved in DC Voltage Amplifier

V-DIV	ESTABLISHES THAT	$V_0 = F \times V_{CC}$
	V_B IS IDENTICALLY V_0	
Q	ESTABLISHES THAT	$V_E = V_B - .7$
	V_{R_E} IS IDENTICALLY V_E	
OHM	ESTABLISHES THAT	$I_{R_E} = V_{R_E} / R_E$
KCL	ESTABLISHES THAT	$I_E = I_{R_E}$
Q	ESTABLISHES THAT	$I_C = I_E$
KCL	ESTABLISHES THAT	$I_{R_L} = I_C$
OHM	ESTABLISHES THAT	$V_{R_L} = I_{R_L} \times R_L$
KVL	ESTABLISHES THAT	$V_{OUT} = V_{CC} - V_{R_L}$

Figure 17

all the components of the novel physical instantiation, the understanding system is able to know when its knowledge base is incomplete, if some components do not appear in any viewpoint explanation. It can inquire about the missing concept and back up the question by referencing the unused components. It also knows what questions to ask based on the meta-level knowledge [6] it has defining its knowledge sources. Furthermore, once a new concept is actively solicited, the understanding system can test the completeness and correctness of its understanding by trying to rationalize the concept on the novel physical instantiation which inspired the system's curiosity, much as a human student would.

The output of the Behavior Verification process is the set of completed viewpoints which each explain one abstract characteristic causally in terms of the physical structure which achieves it. The collection of viewpoints, which comprise the understood mechanism, is passed to the Experience Incorporation process. Behavior Verification has explained the novel physical instantiation in terms of its understood physical substructures.

3.5. Experience Incorporation

The Experience Incorporation process has several responsibilities. It must coordinate the set of viewpoint explanations into a form suitable to respond to directed questions, either stand-alone or from various application expert systems. Since the viewpoint is the basis from which the understanding system focused its understanding efforts, the viewpoint is also the basis of focused explanation. Since the relationship among nodes of the Component-mechanism Hierarchy is one where the father node is explainable in terms of its son nodes (the collection of son nodes in its connection scheme is an instantiation of the abstract father node), the explanation of the novel physical instantiation can take place at various levels of conceptual detail [Figure 18]. For example, the understanding system can explain the DC voltage amplifier in terms of the bias and gain; it can explain the bias viewpoint of the DC voltage amplifier using the voltage divider as a component; it can explain the factor viewpoint of the voltage divider using the resistors R1 and R2 as components; it can explain the bias viewpoint of the DC voltage amplifier using the resistors R1 and R2 as components. The level of explanation should proceed from the highest level of the explanation hierarchy and filter down if concepts such as voltage divider are not familiar to the questioner. This is the power of a conceptual explanation hierarchy, as opposed to the myopic, static, single level explanation offered by production systems such as MYCIN[1].

Another responsibility of the Experience Incorporation is the process of self-extension by properly hooking the understood physical instantiation into the Skill Knowledge Base and propagating the effects of the experience it represents. Specifically, this means the effects of applying the induction paradigm [7, 8, 9] to perturb the Semantic Template Hierarchy. In viewing the understood instantiation as a positive example of the physical concept which is embodied in a Semantic Template, the induction paradigm directs a perturbation of the characterization of the physical concept to include the novel physical instantiation.

The effect of properly hooking the understood physical instantiation and applying the induction paradigm on the Semantic Template

Explanation in Conceptual Vocabulary

$$\begin{tikzcd}
& \underline{\text{DC Voltage Amplifier}} \arrow[dl] \arrow[dr] & \\
\text{Bias Viewpoint} \arrow[d] & & \text{Gain Viewpoint} \\
\text{Rationalized explanation (with } \underline{\text{V-div}} \text{ as component)} \arrow[d] & & \cdots \\
\text{Factor Viewpoint} \arrow[d] & & \\
\text{Rationalized explanation (with } \underline{R_1} \text{ and } \underline{R_2} \text{ as components)} & &
\end{tikzcd}$$

Figure 18

Hierarchy is to extend the Skill Knowledge Base or experience. The understanding system matures as it is exposed to more and more novel physical instantiations, leading to Skill Knowledge Base growth. The greater experience is reflected in the Physical Conceptualization phase of understanding, where novel physical instantiations encountered in future sessions are much more simplifiable. More complex substructures can be composed and viewed as single components because they have been encountered in the understanding system's past experience.

The accumulation of experience brings up a key point. The understanding system can be viewed as a learning system to the extent that it learns novel physical instantiations which it hooks into its Skill Knowledge Base. The learning it performs is by conceptually directed analysis. The learning it performs is supervised [10] in the sense that all the novel physical instantiations are well-designed, named mechanisms. In line with supervised learning, it seems reasonable to regard the exercising of the understanding system as a continuous training sequence. The complexity of the novel physical instantiations should initially be fairly simple and grow increasingly more complex at a moderate pace. For example, before exercising the system with the DC voltage amplifier which includes a voltage divider, the system should be exposed to several voltage amplifiers from which it can progressively refine the corresponding Semantic Template. It can then expediently recognize the voltage amplifier in the Physical Conceptualization phase of rationalizing the DC voltage amplifier.

4. References

[1] Shortliffe, E., Computer-based Medical Consultation: MYCIN, Elsevier Computer Science Library, 1976.

[2] Brown, A., Qualitative Knowledge, Causal Reasoning, and the Localization of Failures, M. I. T. Ph. D. dissertation, 1976.

[3] Lai, K., Functional Testing of Digital Systems, C. M. U. Ph. D. dissertation, 1981.

[4] De Kleer, J., Causal and Teleological Reasoning in Circuit Recognition, M. I. T. Ph. D. dissertation, 1979.

[5] Stallman, R., and Sussman, G., "Forward Reasoning and Dependency-directed Backtracking in a System for Computer-aided Circuit Analysis," M. I. T. Technical Report Memo 380, 1977.

[6] Davis, R., "Interactive Transfer of Expertise: Acquisition of New Inference Rules," Proceedings 6th International Joint Conference on Artificial Intelligence, 1979.

[7] Mitchell, T., "Version Spaces: A Candidate Elimination Approach to Rule Learning," Proceedings 6th International Joint Conference on Artificial Intelligence, 1979.

[8] Michalski, R., "Inductive Learning as Rule-guided Generalization and Conceptual Simplification of Symbolic Descriptions," Workshop on Current Developments in Machine Learning, 1980.

[9] Quinlan, J., "Inductive Inference as a Tool for the Construction of High-performance Programs," Workshop on Current Developments in Machine Learning, 1980.

[10] Smith, R., et. al., "A Model for Learning Systems," Proceedings 6th International Joint Conference on Artificial Intelligence, 1979.