IDT: AN INTELLIGENT DIAGNOSTIC TOOL

Hal Shubin
John Wade Ulrich

Knowledge Engineering Group
Digital Equipment Corporation
South Lawrence, MA 01843

ABSTRACT

IDT is an intelligent hardware diagnostic tool that has been successfully used to identify faults in PDP 11/03 computers. It selects and executes tests, and interprets the results. IDT is able to modify its test selection strategy on the basis of results of previous tests as well as opinions offered to it by the user. Symbolic formulas are used to represent the relationship between the test results and the broken equipment. If we assume that there is only one broken component, then set operations can be shown to be sufficiently general for combining the results of multiple tests.

I INTRODUCTION AND OVERVIEW

The diagnosis of broken computing equipment is an intellectually challenging activity requiring a technician to maintain and to reason about a large collection of related facts. These facts are obtained directly by observation or indirectly by interpretation of the results of diagnostic tests. The technician must recognize when the facts implicate an identifiable part of the computer, called a field replaceable unit (FRU). When the known facts are insufficient, the technician chooses a diagnostic test that is likely to establish facts leading to the identification of the broken FRU.

IDT is a tool designed to help a technician identify which FRU should be replaced to fix the broken computer. It can run in either an automatic or an interactive mode. In its automatic mode, IDT selects, runs and interprets the results of diagnostic tests. When running interactively, IDT assumes a more subordinate role of making suggestions and summarizing results. IDT may move freely between these two modes of operation as the user's experience or needs dictate. In both modes, the technician is kept informed of the progress of the diagnosis by means of a display which is updated as information is gathered.

Previous diagnostic tools (eg, Digital Equipment Corp.'s Techmate [3]) were unable to dynamically alter their test selection strategies. These systems used a predetermined testing order that guaranteed that the preconditions of each diagnostic test were satisfied before the test was run. They could not use information gleaned from test results or from users' expressed opinions to alter the search. As a result, many tests were run needlessly.

Because IDT represents knowledge as symbolic formulas, it is able to perform hypothetical reasoning. This ability allows it to run tests without first determining that all preconditions are met. Thus, it mirrors the behavior of a human diagnostician who will frequently assume that certain parts of the broken equipment are working correctly so that tests which depend on these parts may be interpreted.

IDT is also able to alter its testing strategy on the basis of user-introduced opinions. It does this by combining opinions expressed by the user with facts obtained from previous tests to predict which test is most likely to identify the broken part. Opinions that are inconsistent with the known facts are identified. When inconsistencies are detected, the user is suitably advised and the diagnostic process continues.

IDT is implemented in FRANZ LISP [1] and OPS5 [2]. The program runs on a VAX 11/780 at a site remote from the broken equipment. The current implementation uses an on-site PDP 11/03 to down-line load diagnostic tests. The diagnostic tests used are Digital Equipment Corporation's XXDP+ series of function/logic tests. The VAX and the PDP 11/03 talk over a 4800-baud communications line. IDT has been used with good results to diagnose faults in the RX02 floppy disk subsystem on a PDP 11/03.

II AN ANNOTATED EXAMPLE

The reader may get a good idea of the capabilities and basic concepts of IDT by working through an example. The following is taken from an actual run in which IDT correctly diagnosed an error in an RX02 system which was inserted by shorting the chip that allows the system to detect that the diskette is spinning. We provide only enough detail to understand the example. A complete technical description of IDT will be provided in subsequent sections.

Figure 1 illustrates IDT's current physical configuration. Two computers, besides the unit

under test (UUT), are involved. One is remote and one is local to the UUT. The remote computer, a VAX 11/780, contains the knowledge base, reasoning mechanisms and testing strategies. The local computer, a PDP 11/03, contains the diagnostic test series and the display software. The two computers communicate over a 4800-baud telephone line.

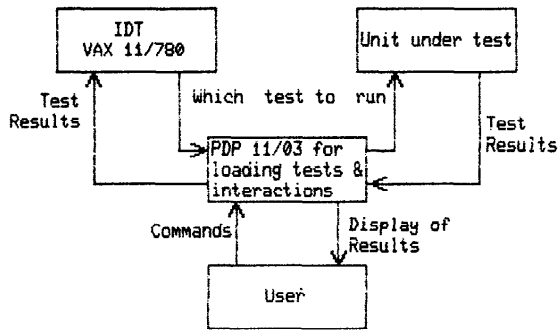

Figure 1: Current Configuration of IDT

The user initiates a diagnostic session by powering up the 11/03, which then telephones the 11/780 and logs in to a special account, initiates the user's display and then retires to a passive role. Future activities of the diagnostic process are controlled by the 11/780 with the 11/03 passing messages, loading and running tests when told to do so, and managing the display.

Figure 2 shows the beginning of a diagnostic session. The screen is divided into three regions. The top region is a menu from which the user may select the next operation to be performed by IDT. The middle section contains a description of the current activity of IDT. The bottom section displays the state of the diagnostic process. After the final menu item is the instruction "enter choice". This informs the user that he may now make a menu selection. The user has requested menu option 2 which allows him to tell IDT which piece of equipment he thinks may be broken.

In Figure 3, the system has responded to the user's request to enter an opinion by displaying the message "enter opinion". The user responded by typing "(read-write)". By typing "(read-write)" the user has indicated that he believes that the read-write board may be faulty and that IDT should try to run tests that will verify this opinion. The user could have typed "(not read-write)" to indicate that IDT should base its testing strategy on the opinion that the read-write module is not broken. Combinations of opinions can also be accepted.

In addition to the read-write unit, there are four other FRUs in the RX02 subsystem. They are: interface, controller, disk-drive (the mechanical portion of the sub-system) and diskette (any floppy disk that is in the drive being tested). The complete list of the FRUs is displayed in the lower third of the screen. Associated with each FRU is a set of atomic "functional" units (AUs). These atomic units constitute the parts of the FRUs that can be tested by the diagnostic tests. The diskette and disk-drive have only one functional subpart, namely themselves.

```
                    IDT Diagnostics

1. SELECT a test                    5. Display HELP information
2. Enter an OPINION                 6. STOP this run
3. Allow program to CHOOSE          Enter choice
4. Show STATUS                      >> 2

-------------------------------------------------------------------

Initializing tables.
Ready to start now.
Demonstration of the IDT project being developed
in the KEG group at Digital Equipment Corp., in
Tewksbury, MA.

-------------------------------------------------------------------

FRU           Atomic Units suspected (empty => NOT suspected.)
interface     int1 int2 int3 int4 int5
controller    ctl1 ctl2 ctl3 ctl4 ctl5 ctl6 ctl7
read-write    rw1 rw2 rw3 rw4
diskette      diskette
disk-drive    disk-drive
```

Figure 2: IDT's Initial Display

```
                    IDT Diagnostics

1. SELECT a test                    5. Display HELP information
2. Enter an OPINION                 6. STOP this run
3. Allow program to CHOOSE          Enter opinion
4. Show STATUS                      >> (read-write)

-------------------------------------------------------------------

Adding your opinion to the tables.
Selecting a test based on your opinion.
  Your opinion caused test 17 to be selected.
Analyzing the result of the test.
  Test 17 failed.

Nothing to report about:    interface, read-write, controller
The tests have eliminated:  diskette, disk-drive

-------------------------------------------------------------------

FRU           Atomic Units suspected (empty => NOT suspected.)
interface     int2
controller    ctl1 ctl2 ctl3 ctl4 ctl5 ctl6 ctl7
read-write    rw1 rw2 rw3 rw4
diskette
disk-drive
```

Figure 3: User's Opinion Processed

As a result of giving IDT the opinion that the read-write module is broken, IDT has selected and run test 17. After analyzing the results of a test, IDT removes from the display the AUs that have been demonstrated to function properly. In this case they are int1, int3, int4, int5, diskette, and disk-drive. The fact that the diskette and disk- drive have no more untested atomic units associated with them implies that they are functioning correctly; that is, they contain no AUs that are suspected. Notice that the interface has almost been eliminated from suspicion but that it still contains one atomic unit that might be broken.

In Figure 4, the user has indicated that IDT should select and run another diagnostic test. The middle section of the display indicates that IDT has selected and run test 24. In this case, the choice of test 24 was based on the user's opinion, and the result of test 17. As a result of the outcome of test 24, all atomic units except ones associated with the read-write unit have been identified as working correctly. Since the only atomic units that may be broken belong to the read-write unit, it is the read-write module that must be replaced.

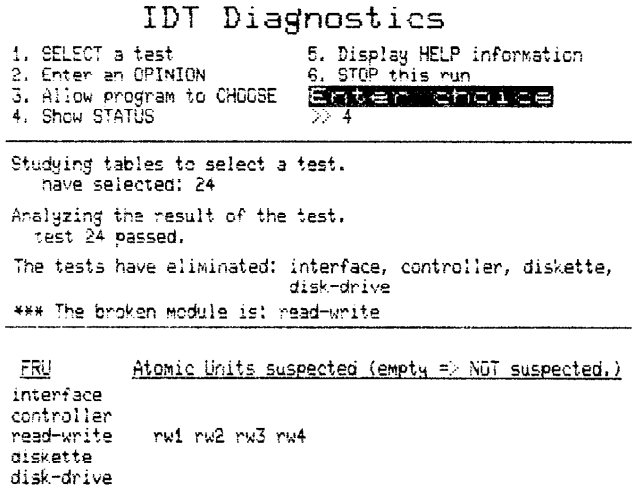

Figure 4: Diagnosis Complete

There are several user options that have not been illustrated by the above example. The options whose meanings are not obvious are 1, and 4. Option 1 is probably only useful for users that are very familiar with the set of diagnostic tests used by IDT, for it allows the user to specify that a particular test should be run next. Option 4 causes IDT to display all previous test results and opinions.

III FORMULATION OF THE DIAGNOSTIC PROCESS

IDT must perform two basic functions:

- It must analyze the results of the tests to determine which FRU should be replaced.
- It must select diagnostic tests from a set of tests. Test selection is based on the knowledge acquired from previous tests and from opinions entered by the user.

To accomplish these functions IDT has:

- a method for interpreting the results of running diagnostic tests
- a method for reasoning about interpreted test results
- a strategy for selecting the testing order.

The remainder of this section is divided into three parts, each devoted to one of the above topics.

A. INTERPRETATION OF TEST RESULTS

To interpret test results, an adequate model of the unit under test must be developed. This is done by studying the diagnostic tests and the block diagrams of the equipment to determine the FRUs of the unit under test and the atomic units of each FRU. For our purposes atomic units are the smallest subpart of the UUT that can be distinguished from other parts by the set of diagnostic tests. An AU may be any part of the UUT (eg, a motor, a chip, circuitry, cabling). We require that no AU belong to more than one FRU. Without this requirement, it would sometimes be impossible to distinguish between two FRUs.

If the diagnostic tests exist before the model is created, then the atomic units are derived by examining the diagnostics. If diagnostic tests do not already exist, then atomic units may be selected first and diagnostic tests then written to distinguish among them. In the latter case, care must be taken to pick AUs of appropriate size. If the atoms are too large, then the test programs will be overly complex. If the atoms are too small, the diagnostic tests required will be too numerous and difficult to interpret.

Knowledge about atomic units is represented by logical formulas. The set of atomic formulas is formed by associating with each atomic unit an identifying symbol. Each such symbol is interpreted as asserting that the associated atomic unit is broken. Non-atomic formulas are formed by combining atomic formulas with the usual set of logical connectives. For example, if A and B are atomic formulas associated with the atomic units x and y, then the formula (A V B) is taken to mean that either unit x or unit y is broken.

To facilitate reasoning about test results, it is useful to require that a test have only two possible outcomes, called "pass" and "fail". If x is a test, then we associate with x two formulas, PASS(x) and FAIL(x). These formulas represent the knowledge obtained when test x is run.

Each of the formulas used to represent test results has the form P => R where P is a statement called the precondition of the test, and R is a formula called the result. The formula P represents the condition that must be true in order that the test result be interpreted correctly, and the formula R represents the condition that is implied by the interpreted outcome of the test.

The advantage of this interpretation of test results is that it permits tests to be run without first determining whether the preconditions are satisfied. This is important since the preconditions may often be verified indirectly by combinations of other test results. Also, precon-

ditions of tests may be used to guide the testing strategy. For instance, suppose that a particular test has the following two formulas associated with it:

$\neg A \Rightarrow C \vee D$ (test failed)

$B \Rightarrow \neg C$ (test passed)

(Remember that "A" means that A is broken and "$\neg$A" means that A is not broken). The work required to verify ($\neg$A) before running the test may be wasted in the case that the test passes. If instead we run the test with no concern for the preconditions we may find out whether we wish to verify ($\neg$A) or B. Thus, the yet-unverified preconditions of a test result may be used to guide further testing.

The formulas obtained from running tests are combined to form statements about FRUs. Initially each FRU, say x, is mapped onto a pair of formulas: INSIDE(x) and OUTSIDE(x). These formulas represent the atomic units that belong to x and those that belong to FRUs other than x, respectively.

As tests are run, the formulas representing the results are conjoined with all of the other formulas in the system and the resulting formulas are simplified. Thus, results from tests affect what is known about the broken equipment as well as what information will be obtained from running new tests. Outcomes of tests become predictable on the basis of previously run tests if their outcome formulas simplify to true or false. It is obviously unnecessary to run such tests since their results can add no new information to the system. When INSIDE(y) simplifies to truth, then the FRU y is known to contain a broken atomic unit.

B. REASONING ABOUT TEST RESULTS

If we interpret an FRU as a set of atomic units, then the goal of the diagnostic process is to prove a formula of the form:

$$A_1 \vee A_2 \vee \ldots \vee A_n$$

where the set $\{A_1\ A_2 \ldots A_n\}$ is a subset of the atomic units on a single FRU. In general, such a proof requires propositional theorem proving about formulas that could contain thousands of atoms. To reduce the complexity of the problem, we make the extra-logical assumption that there is only one broken atomic unit. This "single-fault assumption" is frequently made by technicians when diagnosing broken equipment, and has been used by other diagnostic projects [4, 5]. In terms of the logical formulas described above, the single fault assumption takes the following form:

Assumption: There is exactly one true atomic formula.

In addition to the single fault assumption, there is another characteristic of our formulas that we now make explicit:

Assumption: There is a finite set of atomic formulas.

It should be obvious that this assumption applies in our situation since there are a fixed finite number of atomic units.

The following theorems show that these assumptions greatly reduce the complexity of the theorem-proving problem. Theorem 1 shows that each of our formulas can be represented as a set of positive literal formulas. Theorem 2 shows that set intersection is adequate for conjoining formulas represented as sets of positive literals. Theorems 2 through 5 form the basis for an algorithm that will convert arbitrary formulas into their representative set of literals. In each of the theorems we assume that there are n possible atomic formulas and that exactly one atomic formula is true.

Theorem 1: If E is a formula, then there exists a disjunctive formula containing only positive literals (un-negated atoms) that is equivalent to E.

Proof: Suppose that $E(L_1 \ldots L_n)$ is a formula where $L_1 \ldots L_n$ is the set of all positive literals. (Not all literals may actually occur in the formula $E(L_1 \ldots L_n)$.) Let $E(L_i{=}t)$ denote the result of substituting in the formula $E(L_1 \ldots L_n)$ the logical values t for L_i, and f for L_j for each $j \neq i$. By the single fault assumption, the formula $(L_1 \vee \ldots \vee L_n)$ is identically true. Therefore the following formulas are each equivalent:

1. $E(L_1 \ldots L_n)$
2. $E(L_1 \ldots L_n) \wedge (L_1 \vee \ldots \vee L_n)$
3. $(E(L_1 \ldots L_n) \wedge L_1) \vee \ldots \vee (E(L_1 \ldots L_n) \wedge L_n)$
4. $(E(L_1{=}t) \wedge L_1) \vee \ldots \vee (E(L_n{=}t) \wedge L_n)$

Since $E(L_i{=}t)$ evaluates to t or f for each i, formula 4 is equivalent to a disjunctive formula containing only positive literals.

Clearly, each formula, say E, can be represented by a set of positive literals (namely those positive literals in the disjunctive formula equivalent to E). We use the notation $|E|$ to denote this set of literals.

Theorem 2: If A and B are formulas then $|(A \wedge B)| = |A| \cap |B|$

Proof: Let A and B be two formulas and let $|A| = \{a_1 \ldots a_n\}$ and $|B| = \{b_1 \ldots b_k\}$. Then from theorem 1 plus repeated application of the distributive laws we obtain

$|(A \wedge B)| =$
$\quad |(a_1 \wedge b_1) \vee \ldots \vee (a_i \wedge b_j) \ldots (a_n \wedge b_k)|.$

But by the single fault assumption, $(a_i \wedge b_j) = f$ if $a_i \neq b_j$ and $(a_i \wedge b_j) = c$ if $a_i = b_j = c$. From this it follows that c belongs to $|(A \wedge B)|$

iff c belongs to $|A| \cap |B|$.

Theorem 3: If A and B are formulas then $|(A \vee B)| = |A| \cup |B|$.

Proof: By the associative law.

Theorem 4: If A is a formula and U the entire finite set of atomic formulas, then $|\neg A| = U - |A|$.

Proof: let $|A| = \{a_1 \ldots a_m\}$. Then by theorem 1, and DeMorgan's law $\neg A = (\neg a_1 \wedge \ldots \wedge \neg a_m)$. But the single fault assumption implies $(\neg a_1 \wedge \ldots \wedge \neg a_m) = (b_1 \vee \ldots \vee b_k)$ where $\{b_1 \ldots b_k\} = U - \{a_1 \ldots a_m\}$. The result follows.

Theorem 5: If A is an atomic formula then $|A| = \{A\}$

Proof: Obvious.

If the equalities in theorems 2 through 5 are interpreted as rewrite rules, they can be iteratively applied to convert arbitrary formulas into representative sets of positive literals. Since the computational complexity of the set operations is n squared, the computational cost of such a conversion is reasonably small. Also, since set intersection is the only operation necessary for combining test results, the cost of integrating the results of tests is at worst n squared.

If two faults can occur, we can transform all formulas into another set of equivalent formulas whose atoms are chosen from the cross product of the original set of atomic formulas. If, in this transformed set of formulas, the atomic unit x corresponds to (y, z) in the original, we choose the formulas so that x is true if and only if y and z are true in the original. The single fault assumption now holds for this new set of formulas. This process can be iterated for greater numbers of possible faults. If k is the number of possible faults, then the cost of combining test results grows as $(n^k)^2$ where n is the number of atomic units.

C. TEST SELECTION

In addition to reasoning about test results, IDT can select and initiate new tests. To accomplish this, IDT uses statistical information that predicts the likelihood that a particular AU will be broken, opinions offered to it by the user, and results of previous tests.

To process such information, IDT maps each test, say x, onto two formulas, PASS*(x) and FAIL*(x). Initially, PASS*(x) and FAIL*(x) are identical to the functions PASS(x) and FAIL(x).

As the user gives opinions in terms of logical statements about FRUs (see the example in Section 2), these statements are mapped via the functions INSIDE and OUTSIDE onto statements about atomic units. The resulting formulas are then conjoined with each PASS* and FAIL* formula. It is possible that both PASS*(x) and FAIL*(x) are false. This would indicate an inconsistency in the opinions presented to IDT or a violation of the single fault assumption.

Intuitively, the values of PASS*(x) and FAIL*(x) represent the information that the IDT thinks will be obtained when x is run. For instance, if FAIL*(x) = false, then IDT thinks that the test will not fail since this would imply something it thinks is impossible. In this case, no information could be obtained from running test x. In case the user does not wish to offer an opinion, FAIL* and PASS* remain equal to FAIL and PASS respectively.

By combining the set of objects identified by FAIL*(x) and PASS*(x) with the statistical measure of the probabilities that each AU will fail, we get a measure of the probability of each test failing and passing. We can associate with each test x, the expected size of the smallest |OUTSIDE(y)| where y ranges over the set of FRUs. Call this size the preference number of x. The test selection criterion is then:

> Pick the test with the smallest preference number.

This test selection strategy picks tests that drive OUTSIDE(y) to false for some y. Intuitively, the justification for this strategy should be obvious; when OUTSIDE(y) = false, then y is known to be broken. Because of lack of statistical data, all experimentation with IDT has been made with the assumption that all failures are equally likely. However, even with this assumption, the test selection strategy appears to focus quickly on the most likely broken FRU when guided by appropriate user opinions. When guided by incorrect opinions, however, the system can be temporarily mislead.

IV SUMMARY

We have described an intelligent diagnostic tool (IDT) which is demonstrably useful in tracking down bugs in broken computers. It has the following properties:

- IDT uses a rather high-level model of the equipment to be tested.
- If IDT is used with an existing set of diagnostic programs, then the model can be derived from the diagnostics. Otherwise a model must be specified and a set of diagnostic tests created.
- IDT can analyze the results of running diagnostic tests and can select new tests to run based on information provided by previous test results.

- IDT can also accept and process opinions offered to it by a technician. These opinions guide the diagnostic process along lines thought to be promising by the technician.

- IDT may use statistical information to guide its testing strategy so that the diagnostic process will be biased toward frequently occurring problems.

V REFERENCES

[1] Foderaro, J.K., "The FRANZ LISP Manual", University of California, 1980.

[2] Forgy, C.L., "OPS5 User's Manual", Carnegie-Mellon University, 1981.

[3] Kiskiel, M., et. al. "Techmate Functional Specification", Digital Equipment Corp., 1981.

[4] Tendolkar, N.N., and R.L. Swann, "Automated Diagnostic Methodology for the IBM 3081 Processor Complex", IBM Journal of Research and Development, 26:1 (1982), 78-88.

[5] Davis, K.R., et. al., "Diagnosis Based on Structure and Function", Proceedings of the 1982 National Conference on Artificial Intelligence, 1982.