A RESOURCE ALLOCATION ANALYZER FOR SOURCE PROGRAMS WRITTEN IN ATLAS

Michael T. Mills
U.S. Air Force
Aeronautical Systems Division (ENEGE)

ABSTRACT

This paper describes the design of a computer program which will read a source program written in abbreviated test language for all systems (ATLAS) and furnish a list of resources required by the ATLAS program to successfully conduct a test. These resources include specification requirements of test equipment used to generate or to sense test signals and impedance load requirements needed for the unit under test. This design uses high level data flow analysis of ATLAS (IEEE STD 416-1976) source code to determine the required resources of an ATLAS program.

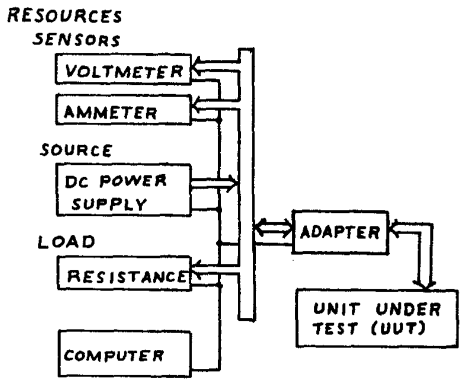

FIGURE 1

INTRODUCTION

This presentation describes a computer program called a resource allocation analyzer for avionics test programs which are written in ATLAS (Abbreviated Test Language for All Systems) as defined in IEEE STD 416-1976. ATLAS is a high order language used to control automatic test equipment (ATE). Maintenance personnel often use ATE to test and checkout complex systems such as avionics or radar fire control systems.

A system or part of a system to be tested is often referenced as the unit under test, or UUT. During a test, the UUT requires signal sources with critical characteristics attached to some of its connector pins while UUT generated signals are measured from other connector pins. An ATE configuration shown in FIG 1 consists of an adapter, various loads, test instruments, and a computer. The UUT is connected to the ATE configuration through an adapter. The adapter selects connector pins called for by the ATLAS program. Loads are used to satisfy certain load requirements required by the UUT on some of its connector pins. A load may include a resistance, a capacitance, an inductance, or some combination of each. Test instruments include sources which generate signals into the UUT and sensors which measure signals from the UUT. The ATLAS program uses the computer to control the adapter, loads, and test instruments during a test. This presentation refers to loads and test instruments, including sources and sensors, as resources.

NEED FOR THIS PROGRAM

When ATE is being constructed and test programs are being developed many problems are encountered during the process. Early in the program there is a need to define the ATE hardware so design and development can begin in order that the equipment can be delivered concurrent with the first aircraft. This requires that test parameters and test times for all Line Replaceable Units (LRUs) be defined and studied so that tradeoffs can be conducted to determine station configurations and loadings. By first determining LRU test parameters (maximum or minimums) and then grouping similar requirements combined with compatible test times an optimum test station configuration that is cost effective can be derived. The stations are then built and the software written. The problem arrives when the detailed program is first put on the station. Because LRU test requirements were based on best data at the time and included only maximum requirements, it is possible that full data was in error or incomplete at the time. An example might be where a requirement exists for a power supply of 2 volts at 28 amperes and another power input was 1.6 volts at .5 amperes. The need for the 2 supplies was foreseen but when the programmer called for the first connection of 1.6 volts at .5 amperes the Test Station assigned the 2 volt, 28 amp supply because it didn't know the next step was going to require the other supply. The resource analyzer will catch all requirements, extracted from the actual program and identify overlap, conflicts and other anomalies.

Second, the programmers may not always be aware of the exact configuration of the test station thus it is possible for a test program to compile for say 5 or 6 hours and then have the program call for either simultaneous assets or non-available assets, at which time the program halts and dumps leaving wasted time. ATLAS programs for complex LRUs may require two or more 25MB disc paks so major losses of this type in time and personnel should not be permitted to occur.

Third, when items are to be tested at depot level, it is always desirable to use existing ATE at the depot. When the test program is in ATLAS, the resource allocation analyzer can provide a screening of program requirements for comparison with the AFLC data bank describing the ATE resources. A match would then provide a cost effective way of testing the unit.

RESOURCE ACCUMULATION

The data flow analysis routine performs a depth first search of a flowgraph representing an ATLAS program. Figure 2 contains examples of such a flowgraph. The nodes, which are circles in the flowgraph, represent starting points, end points, control statements such as conditional branches. destination points, and other reference points in the program. Path segments between each node and successor node in the flowgraph contain letters which represent resources required by the ATLAS program.

The data flow analysis program can traverse flowgraphs with parallel tasks (Figure 2b) and loops (Figure 2c). However, when a loop is detected, the resources required are based on one traversal of that loop. No attempt is made to determine the number of times a loop is executed. In some cases, a variable which determines the number of times a loop is executed is not known until execution time. Two marking variables are used to detect parallel tasks and loops.

No path in the flowgraph is traversed more than once. The number of resources required by any section of the flowgraph is saved until the depth first search is completed. This feature prevents any recalculation of resources for any section of the flowgraph which involves parallel

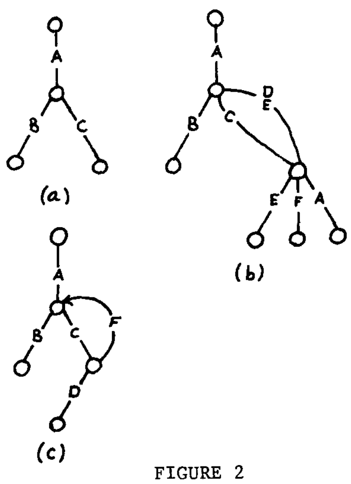

FIGURE 2

tasks. It also prevents loops from being traversed continuously.

Any resources which occur in the flowgraph in series are "added" and any resources which occur in parallel are "or" ed. For each node of the flowgraph, tables are used to calculate the maximum required resources. One table called PATH-RES contains resources required in a path segment between a node and its predecessor node. Another table called WORST-PATH contains the maximum number of resources encountered from a node through any of its successor paths. Whenever PATH-RES > WORST-PATH, WORST-PATH is made equal to PATH-RES. Both of these tables contain a separate row for each type of resource. For example, the number of DC POWER SUPPLIES are calculated in a row separate from the number of DC VOLTMETERS. Figure 3 shows an example flowgraph with a separate set of tables for each node. The WORST-PATH table located at the root of the flowgraph contains the worst case number of resources required by the corresponding ATLAS program.

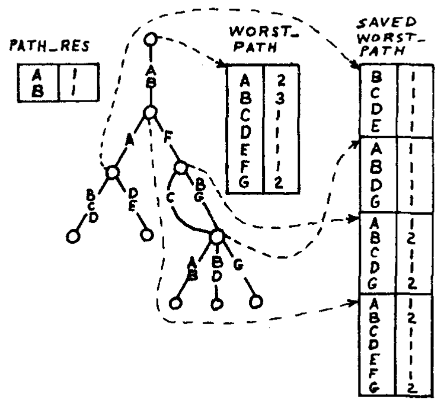

FIGURE 3

DATA STRUCTURES

A variety of data structures perform the following functions:

1. Save information about statements which effect resources.

2. Detect and distinguish between different kinds of resources.

3. Hold detailed resource specifications.

4. Show the effects of ATLAS control statements.

Figure 4 shows the data structures which represent the flowgraph and resource effecting statements. Each row in the flowgraph table represents a node in the flowgraph. Included in each row is a linked list of elements. Each element represents a possible path that the program can take from that node. This linked list allows any node to have an unlimited number of possible paths. Each element in the linked list contains a separate field for the following items:

1. The number of statements in the path segment represented by this element.

2. A pointer to the represented block of statements in the statement table.

3. A pointer to the son node in this path.

4. A pointer to the next element of the linked list.

Each block of statements in the statement table represents statements which are located between node producing statements in a program.

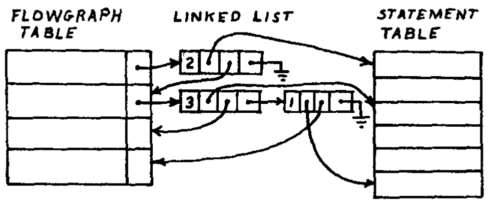

FIGURE 4

DETECTING AND DISTINGUISHING RESOURCES

Resources are specified in ATLAS statements by certain verbs, nouns, noun modifiers, and statement characteristics and are accessed through connector pins. The following source code is an example of an ATLAS statement involving a resource.

```
100205 APPLY, DC SIGNAL, VOLTAGE 10V ERRLMT
+ - 0.1V, AC-COMP MAX 10MV, CURRENT MAX 0.2A,
CNX HI J1-1 LO J1-2 $
```

Data structures are created from the ATLAS source code by a parser and are used to detect and distinguish between resources. In Figure 5, two data structures called verb-noun arrays (a) and (b) are created from verbs such as APPLY and nouns such as DC SIGNAL. The verb and noun combination "APPLY, DC SIGNAL" specifies that a signal source is required and a DC POWER SUPPLY must generate that signal. Therefore the resource required by this example ATLAS statement is a DC POWER SUPPLY. Verb-noun array (a) contains pointers to resource names in a list of standard resources. Verb-noun array (b) contains pointers to a resource table. This resource table is the WORST-PATH resource table discussed above. A linked list column of the resource table links together the resources which are determined by a verb-noun combination.

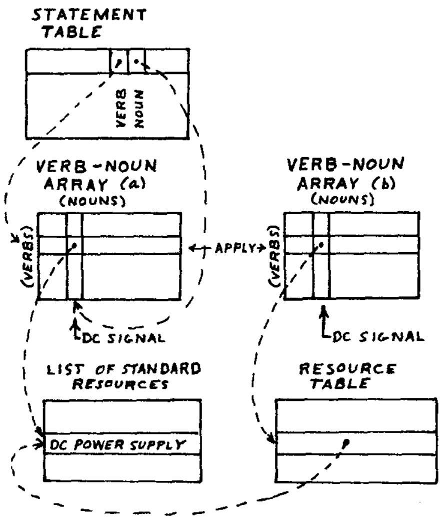

FIGURE 5

Once a resource effecting statement in the statement table is reached during the depth first search, the following actions take place. The verb and noun pointers in the present row of the statement table are used as indices to the verb-noun array (a). The pointer which is found in the location designated by the DC SIGNAL is used to locate the resource name "DC POWER SUPPLY" associated with this verb-noun pair. If a pointer is absent in the corresponding element of verb-noun array (b), then a pointer to "DC POWER SUPPLY" in the list of standard resources table location is inserted into the corresponding verb-noun array (b). In this case, this is the first time that this resource has been used. However, if this is not the first use of this resource, then a linked list of resources which have been referenced by this verb-noun pair must be searched. This search is required to distinguish this resource from other resources in greater detail. Other data structures need to be accessed before this greater detail is found.

USING DETAILED RESOURCE SPECIFICATIONS

Additional portions of the above ATLAS statement give information on the specifications of the dc power supply. "VOLTAGE 10V" means that 10 volts is required from the dc power supply. "ERRLMT + - 0.1V" means that this 10 volts must be accurate to within plus or minus 0.1 volts. The word "VOLTAGE" is a noun modifier. The symbol 10V with ERRLMT + - 0.1V is a characteristic of that noun modifier and is called a statement characteristic. Figure 6 shows that the data structures which contain noun modifier and statement characteristic information. Other noun modifiers with their statement characteristics from the above ATLAS statement include AC-COMP MAX 10MV and CURRENT MAX 0.2A. This means that the power supply should have an AC component no greater than 10 millivolts and be able to supply a maximum current of 0.2 amps. These noun modifiers and corresponding statement characteristics are inserted into the data structures in Figure 6 by the parser. A symbol table contains the numerical values of the statement characteristics. This is done to account for the extensive diversity of data types allowed in ATLAS.

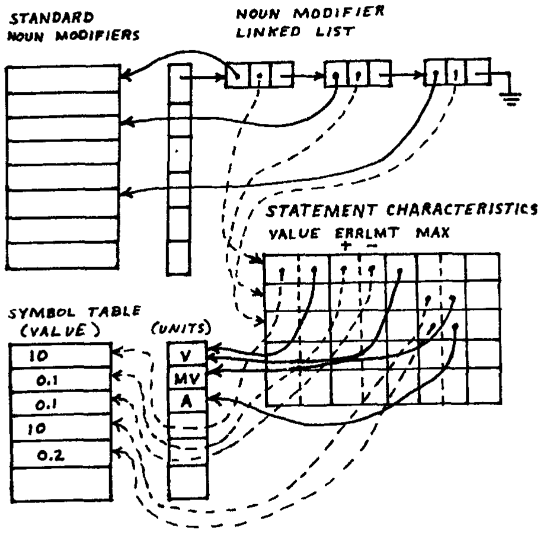

FIGURE 6

REMEMBERING WHERE RESOURCES ARE CONNECTED

From the above example, the phrase CNX HI J1-1 LO J1-2 means this resource is to be connected to pins 1 and 2 of connectors J1. The positive lead (HI) is connected to pin 1 and the negative or neutral lead is connected to pin 2. In order to keep track of what resources are connected to what connector pins at any point in the program, the data structures in Figure 7 are used. A connector matrix contains pointers to resources which are connected to connector pairs represented by the connector matrix. A column in the resource table contains a linked list of resources for each connector pin in the connector matrix which has a connected resource. A pointer inserted into the connector matrix is the result of the above example.

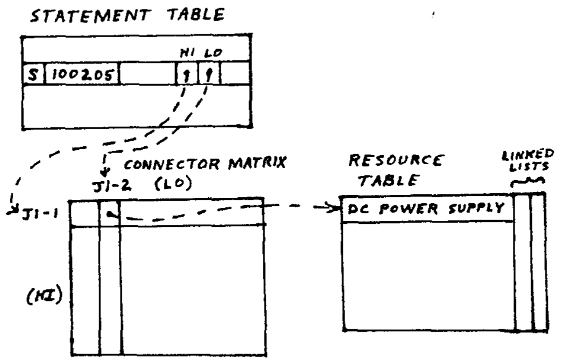

FIGURE 7

The above data structures are only a portion of several data structures used by the resource allocation analyzer. However, the intent of this paper is to show enough data structures to illustrate the approach taken in finding the worst case number of resources required by an ATLAS program.

THE PARSER

A parser reads the ATLAS source program and furnishes the data structures to be used in finding the required resources. This parser is based on an LL (1) grammar which describes enough of the ATLAS language to provide the data structures needed for finding resources. An LL (1) tester is designed to test the grammar for LL (1) by finding its selection sets and showing that they are disjoint. The grammar and selection sets are used to design a pushdown machine for the parser.

It is intended that the resource allocation analyzer will be written in Pascal and will reside initially on the CDC6600/Cyber 73. Standard PASCAL will be used to permit increased transportability. The program will be available for the use of any government contractor developing test programs in ATLAS.

REFERENCES

1. R. Tarjan, "Depth-First Search and Linear Graph Algorithms," SIAM J. Comput., Vol. 1, No. 2, pp. 146-149, June 1972.

2. R. Tarjan, "Finding Dominators In Directed Graphs," SIAM J. Comput., Vol. 3, No. 1, pp. 63-65, March 1974.

3. B. K. Rosen, "High-Level Data Flow Analysis," Communications of the ACM, Vol. 20, No. 10, pp. 713-714, October 1977.