Embedding CLIPS in a Database-Oriented Diagnostic System

Tim Conway

Senior Engineer, Research & Development
Allied-Signal Aerospace Company
Bendix Test Systems Division, MS 4/8
Teterboro, N.J. 07608

0. Abstract

This paper describes the integration of CLIPS into a powerful portable maintenance aid (PMA) system used for flightline diagnostics. The current diagnostic target of the system is the Garrett GTCP85-180L, a gas turbine engine used as an Auxiliary Power Unit (APU) on some C-130 military transport aircraft. This project is a database oriented approach to a generic diagnostic system. CLIPS is used for "many-to-many" pattern matching within the diagnostics process. Patterns are stored in database format, and CLIPS code is generated by a "compilation" process on the database. Multiple CLIPS rule sets and working memories (in sequence) are supported and communication between the rule sets is achieved via the export and import commands. Work is continuing on using CLIPS in other portions of the diagnostic system and in re-implementing the diagnostic system in the Ada language.

1. Project Overview

The purpose of this project is to develop a generic, database-driven, flightline and/or on-board diagnostic system for electronic and electro-mechanical systems. To re-target the diagnostic system to another device, a new diagnostic database would be generated. The diagnostic system is fully integrated with an on-line hypertext technical manual presentation system.

The initial target for the system is the Garrett GTCP85-180L gas turbine engine. This device is used as an auxiliary power unit (APU) on many military and civilian aircraft. It generates electrical power and pneumatic power used to start the main engines and run systems on the aircraft.

The development environment for this project is a SUN workstation, using "C", CLIPS, and a commercial network database product. The target environment is a portable maintenance aid (PMA) prototype developed by Allied-Signal Aerospace Company, Bendix Test Systems Division. The PMA contains a 68030, 25 Mhz microprocessor with 8 Mb DRAM and 2 Mb PROM, and a 34010 40 Mhz graphics co-processor with 2 Mb DRAM and 1 Mb PROM, a 640 x 480 double super-twist LCD display, a special purpose keypad and a removable 3 Mb SRAM cartridge. The PMA runs a multi-tasking operating system, a Bendix developed windowing environment based on the X windows system, CLIPS, and a commercial network database product. A commercial Analog/Digital data acquisition board is installed, with its own 68000 microprocessor and memory. The PMA weighs 10 pounds, and is 3" x 11" x 16". A removable battery pack, if needed, adds 1" in depth, and 10 pounds.

For this application, a signal conditioning unit, cables, and various sensors connect the PMA directly to the APU for on-aircraft, flight line diagnosis. The signals monitored by the diagnostic system consist of digitized analog signals, such as: Exhaust Gas Temperature (EGT), Oil Pressure (POIL), Compressor Discharge Pressure (PCD), Fuel Pressure (PFUEL), shaft rotations (RPM) and a collection of digital control signals.

2. AI Philosophy

Like any other software approach, AI software techniques have their strengths and weaknesses. A general design guideline for this project is to combine a number of approaches in such a way as to use each for the tasks it does best. In this project, CLIPS (and the RETE algorithm) yields advantages in "many-to-many" pattern matching that procedural programming techniques can not deliver. However, CLIPS is not optimal for our application in implementing the consequences of this pattern matching. "C" code and a network database are more suited to this type of task and are used for this purpose in the system. Such things as window management, the user interface, and the data acquisition subsystem are also implemented in "C". As a result, the CLIPS pattern matching code contains only the minimum data necessary to perform its function. In this way optimum performance is achieved.

3. CLIPS Use: Pattern Recognition Within the Diagnostic Software

In this application, (APU maintenance) the data acquisition sub-system provides the diagnostic system with a file of data sample records. Each of these records contains a "snapshot" of all of the analog and digital parameters available for the system at that instant in time. The task of the diagnostic system is to extract from this stream of data samples the significant information they contain. This is a two step process, consisting of Event recognition and Pattern recognition.

An Event is any data condition that can be determined from a single data sample record. A history mechanism (the State Vector) allows knowledge of previously recognized events. By definition, an Event occurs at a specific point in time. An example of an Event would be an overheat condition, where an EGT greater than a set amount in any data sample signals an overheat of the APU. Other examples are: transitions of discrete signals, combinations of discrete signals existing simultaneously, and combinations of discrete and analog signals occurring simultaneously.

A Pattern is a higher level concept, encompassing those conditions descriptive of more than one point in time. Patterns exist in hierarchies and can build upon the existence or absence of lower level Patterns and Events. An example of a Pattern would be a failed-lightoff, where an Event detecting sufficient fuel pressure was found, but no Event detecting combustion was seen. Each Event or Pattern can have an effect on the suspicion levels of system components. The suspicion levels of components are adjusted up or down in response to Event and Pattern recognition.

The primary purpose of Event recognition is data reduction. Each data sample "snapshot" is evaluated against each active Event definition once. If the data sample does not trigger any Events, it is discarded. If the sample triggers at least one Event, a copy of it is kept. In this manner, a large proportion of the data samples are discarded and a small number of significant occurrences in the data sample stream are collected for further analysis.

The RETE algorithm performs best when working memory changes slowly. If CLIPS were used for Event recognition, working memory would consist of a single Data Sample record that would change each time a new Data Sample record was acquired. In this case, the benefits of RETE would not be realized. For this reason procedural "C" code is used for Event recognition.

CLIPS is used for Pattern recognition because working memory consists of a set of Event Recognition (ER) and Pattern Recognition (PR) facts which change much more slowly. New facts are added during Pattern recognition based upon successively higher levels of Patterns building on existing facts. In this case, RETE is a very efficient algorithm to use.

The salience feature of CLIPS is useful here as well. Each Pattern is expressed as one or more CLIPS rules. Patterns are expressed in a hierarchy and each Pattern can include conditions that depend on the absence of another lower level Pattern or Event. Rule salience is used here to enforce the hierarchy so that lower level Patterns, that can produce these Pattern Recognition facts are evaluated before Patterns that are conditional on their absence.

4. Database to CLIPS Compilation

The diagnostic system allows multiple sets of Event and Pattern records. For a given target system there is a "root" database, and a number of "child" databases. The root database is the highest level set of Patterns for the target system. An example of a child database would be the Events and Patterns for analyzing the data samples from an APU start-up.

All diagnostic data in the system is stored in database format. This includes root and child Pattern descriptions. To use CLIPS for Pattern recognition, CLIPS rules must be generated for the Patterns. This is done by a "compilation" process on the database. One or more CLIPS rules are generated for each database Pattern. Each rule expresses the Pattern's criteria in logically equivalent CLIPS syntax. A separate file is generated for the root database Pattern rule set and each of the child database Pattern rule sets. This process is accomplished on the host machine and the CLIPS rule set files become a part of the diagnostic database that is downloaded to the target machine.

There are two types of facts that these rule sets work on. These are Event Recognition (ER) facts, and Pattern Recognition (PR) facts. Both types of facts contain fields identifying the database number and identifying number of the Event or Pattern the fact represents. Each type also contains a unique sequence number to distinguish it from all other facts of its type and to allow multiple occurrences of the same ER or PR to exist, each with different sequence numbers. ER facts will also contain a copy of the analog and digital parameters of the data sample which caused their recognition. The digital parameters follow the analog parameters, with each group of 16 discrete parameters compressed into a single 16 bit integer.

Figure 1 shows the organization of a typical Pattern record, and its related records in the database. Each

Pattern can have multiple Event Criteria (EC) records associated with it. Each of these forms a single logical test which in turn is translated into a single CLIPS Left Hand Side (LHS) condition. An EC record can represent either the existence or absence of an ER fact or PR fact. (i.e. the prior recognition or lack thereof of either an Event or Pattern) For ER facts, additional Parameter Condition (PC) records may be specified to further test the analog or digital parameters. These tests can be against constants, other parameters in the same ER fact, or other parameters in another ER fact under another EC for the same Pattern. This allows conditions such as: "the fuel pressure at the 35% RPM Event is at least 10 psi higher than the fuel pressure at the 10% RPM Event". Parameter Conditions also have tolerance ranges which allow for sensor errors and other types of inexact conditions.

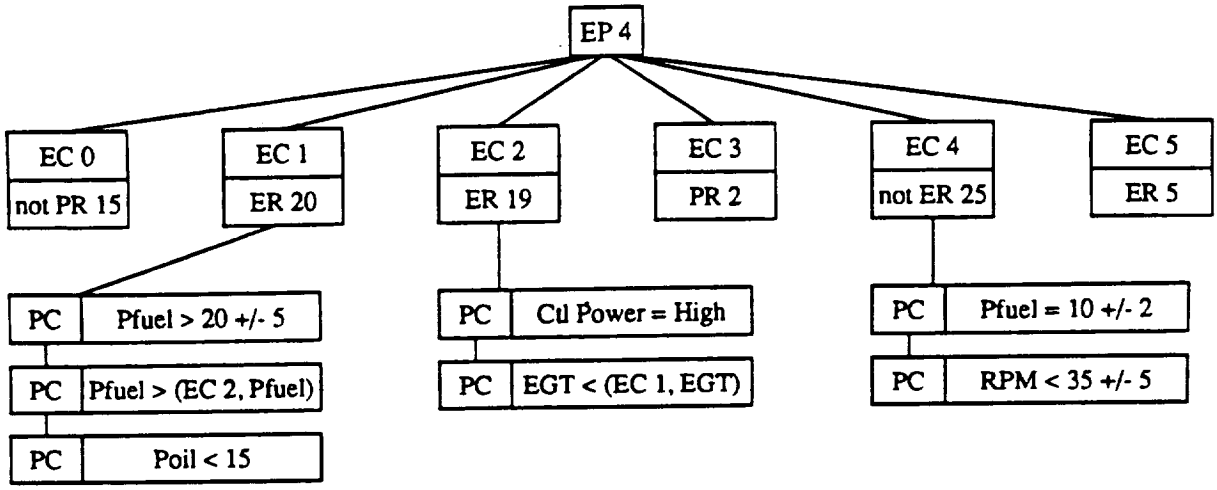

Figure 1 - Typical Pattern instantiation in the database.

Two problems related to CLIPS syntax must be dealt with. First, the first LHS condition of a rule may not be negated. Second, no named fact field can be used in a comparison until after the name has been declared. The first problem can be alleviated by ordering the LHS conditions generated such that non-negated PRs and ERs precede negated ones. This way, if there are any non-negated LHS conditions, they will appear before any negated LHS conditions. While it is still possible for a pathological Pattern to be written with no non-negated ECs, this can be checked for by loading the rule set on the host system after it has been generated. Pattern rules that fail for this reason can then be re-written to add at least one non-negated ER or PR. Figure 2 shows the Pattern from Figure 1 reorganized to move the non-negated ER and PR elements to the start of the rule.

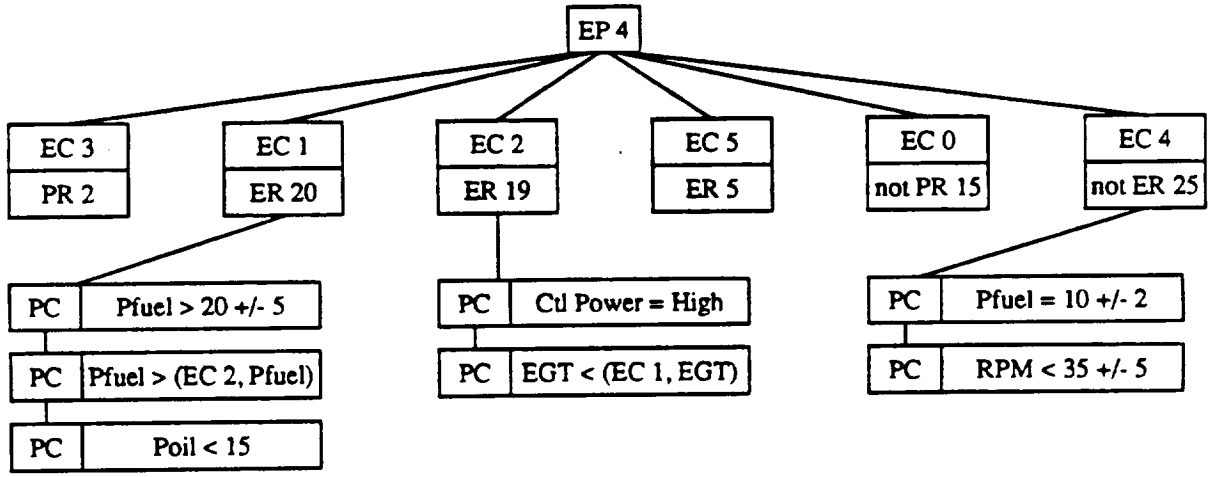

Figure 2 - Pattern from Figure 1 partially reorganized for CLIPS rule compilation.

The solution to the second problem is shown in Figure 3. Since we have now established the order that the EC LHS conditions will be written, we can now go through them looking for "forward references". These can occur

when a Parameter Condition (PC) under an ER fact references a parameter which has not been declared by that point in the compilation. When this occurs, we can "reverse" the condition, and attach it to the forward referenced LHS condition.

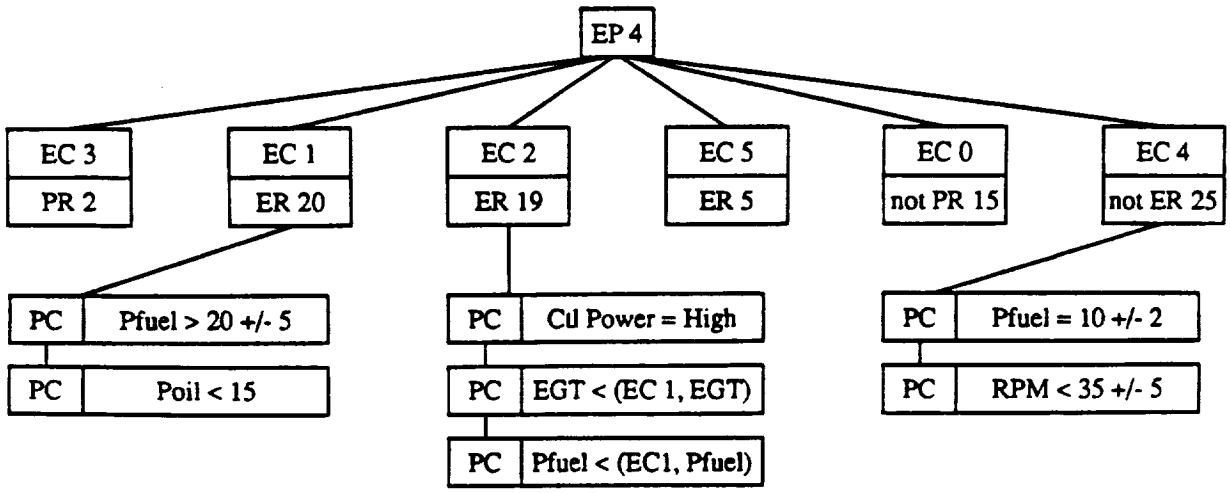

Figure 3 - Pattern from Figure 1 fully reorganized for CLIPS rule compilation.

In Figure 2, a left-to-right compilation of ECs into LHS conditions will fail at the second PC under EC 1. This PC contains a reference to an element (Pfuel under EC 2) that has not been declared yet. To resolve the problem, the condition is reversed, and placed under the forward referenced EC. (EC 2) This is shown in Figure 3. This creates an equivalent logical condition, in a form that CLIPS can digest.

```
; EP # 4 translated

(defrule DB-001-EP-00004 "test rule"
     (declare (salience 290))
     (PR 2 2 $1)
     (ER 2 20 ?ER01 ? ?EC1Ltime ?EC1Time
               ?EC1Poil&:(< ?EC1Poil 15)
               ?EC1Pcd
               ?EC1Pfuel&:(> ?EC1Pfuel 25)
               ?EC1EGT ?EC1RPM
               ?EC1S7d )
     (ER 2 19 ?ER02 ? ?EC2Ltime ?EC2Time ?EC2Poil ?EC2Pcd
               ?EC2Pfuel&:(< EC2Pfuel EC1Pfuel)
               ?EC2EGT&:(< EC2EGT EC1EGT)
               ?EC2RPM
               ?EC2S7d&:(DCc ?EC2S7d 2 1 0))
     (ER 2 5 ?ER05 ? ?EC5Ltime ?EC5Time ?EC5Poil ?EC5Pfuel ?EC5EGT ?EC5RPM
               ?EC5S7d)
     (not (PR 2 15 $?))
     (not (ER 2 25 ?ER04 ? ?EC4Ltime ?EC4Time ?EC4Poil ?EC4Pcd
                    ?EC4Pfuel&:(&& (>= EC4Pfuel 8) (<= EC4Pfuel 12))
                    ?EC4EGT
                    ?EC4RPM&:(< EC4RPM 30)
                    ?EC4S7d)
=>
     (call (NewPR 2 4)))
```

Figure 4 - CLIPS rule generated from compiling EP 4.

Figure 4 shows the CLIPS rule generated from the reorganized Pattern in Figure 3. The rule name is generated from the database number and identifying number of the Pattern. A more descriptive name is placed as a comment next to the rule name. The salience of the rule is computed to match the Pattern's place in the Pattern hierarchy. Next, the ECs under the pattern are each evaluated and categorized as either negated ERs, non-negated ERs,

negated PRs, or non-negated PRs. At this time, the sequence field of each EC's ER or PR fact is given a unique name to allow comparisons of data fields between facts.

Comparisons of analog parameters are done directly in CLIPS code. Comparisons of discrete parameters are done in external routines. "DCc" is an external routine to compare a discrete value with a constant. "DCr" is a similar routine for a relative comparison of a discrete with another discrete. Tolerance ranges are allowed for any comparison. This means that a condition such as "fuel pressure equals 10 psi, +/- 2 psi" would be expressed logically as "((PFUEL <= 12) and (PFUEL >= 8))". Similar condition effects are created for other logical operators.

Lastly, the consequences of each rule are implemented in an external routine, "NewPR". This routine will affect the Ambiguity Group (suspicion) ranking, and will assert a PR fact to indicate recognition of this rule's Pattern.

There are a few additional rules for Pattern construction. For example: relative referenced PCs (i.e. Parameter Conditions under an EC that reference fields in facts other that the fact the parent EC is referencing) are only allowed to reference non-negated ERs. The reason for this is that negated ERs are by definition not present when the rule is evaluated True. Therefore, comparisons of fields of a non-existent fact makes no sense. The compilation process flags these as errors.

5. Database Pattern Sets at Run Time

Figure 5 illustrates the relationship between the root and child databases. The diagnostic system is started with the CLIPS rule set for the "root" database loaded. (i.e. the "rules" file from "root DB" in Figure 5) The root database is the highest level set of Patterns for the target system. When a specific test is to be run, the facts from the CLIPS rule set for the root database are exported to the root database's "facts" file. Then the current CLIPS rules and facts are cleared and the rule set for the selected child database is loaded from its associated "rules" file.

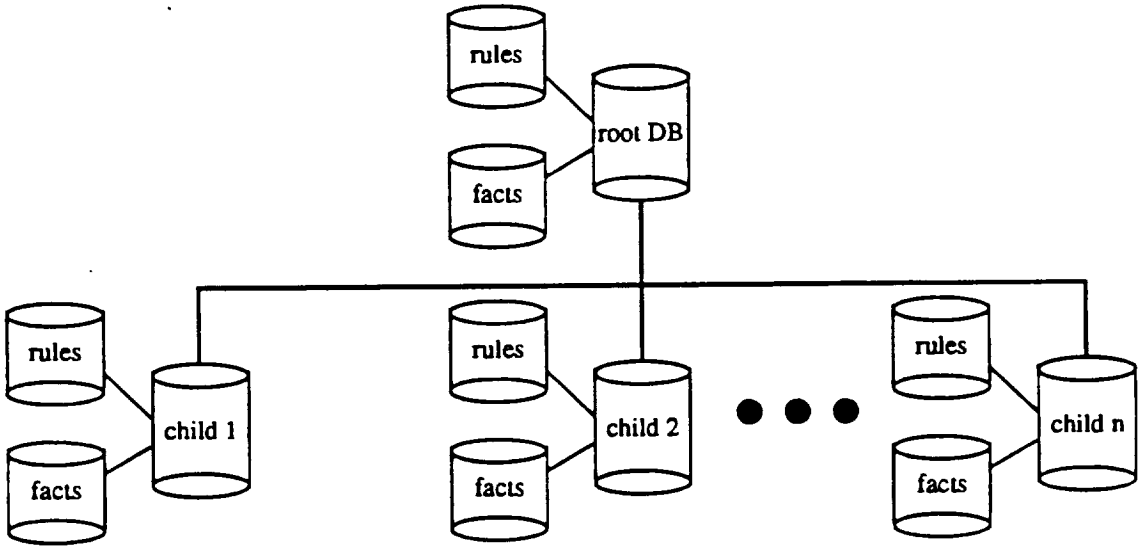

Figure 5 - "Root" and "child" database relationships.

The child Event and Pattern set is then run against the input Data Sample records. Events and Patterns in the child database affect the Ambiguity Group (suspicion levels) in the same manner as root Events and Patterns do. When the conflict set is empty for the run of the child database, Event Recognition (ER) and Pattern Recognition (PR) facts are exported to the "facts" file for the child database. The current CLIPS rules and facts are then cleared and the root database rule set is re-loaded from its "rules" file.

The exported "facts" file from the root database is then re-loaded and a global flag is set to disable the external consequences of rule firings. The root database is then run until the conflict set is empty, causing the root database rule set to return to its last interrupted state without repeating any external effects already accomplished.

Next, the exported "facts" file from the child database is imported and the global flag is reset to allow external consequences of rule firings. The rule set is then run until the conflict set is again empty. In this manner, the root database rules can incorporate additional knowledge from the child database rule firings. This allows still higher

level conclusions to be drawn by the root database about conditions that can not be determined in a single test run. Examples of this would be tracking the degradation of the unit over time or evaluating the results of a calibration or adjustment.

When facts are created by a database, one of the fields in the fact is an "export" flag. When facts are exported, only facts with the export flag set are exported. This allows some degree of purely "local" reasoning to take place in child databases, without burdening the root database with every intermediate conclusion in the child database's reasoning process. Child database Patterns are written with the export flag set only for the "highest" level of Patterns in the child database. These are conclusions that the root database can logically do further reasoning on.

6. Future work

This project is part of a continuing research and development effort to improve flightline and on-board diagnostics and monitoring capabilities for complex electro-mechanical systems. Work is continuing on enhancing the basic capabilities of the system to include such things as better explanation capabilities. CLIPS may be used in other portions of the system wherever it will improve performance and/or capabilities. Work is also underway to re-implement the "C" portions of the system in Ada. A new generation of prototype PMA target hardware is under construction. This will make the system more powerful and easier to embed within a target system.

Work is also progressing on the Knowledge Editor. This is a workstation-based set of tools for analyzing target system data sample sets and creating diagnostic databases. A Script Editor is being written to aid in the creation of the on-line manuals needed for diagnostics and maintenance.