Analysis of MMU FDIR Expert System

Dr. Christopher Landauer
Computer Science and Technology Subdivision
The Aerospace Corporation

April 30, 1990

Abstract

This paper describes the analysis of a rulebase for fault diagnosis, isolation, and recovery for NASA's Manned Maneuvering Unit (MMU). The MMU is used by a human astronaut to move around a spacecraft in space. In order to provide maneuverability, there are several thrusters oriented in various directions, and hand-controlled devices for useful groups of them. The rulebase describes some error detection procedures, and corrective actions that can be applied in a few cases.

The approach taken in this paper is to treat rulebases as symbolic objects and compute correctness and "reasonableness" criteria that use the statistical distribution of various syntactic structures within the rulebase. The criteria should identify awkward situations, and otherwise signal anomalies that may be errors. The rulebase analysis algorithms are derived from mathematical and computational criteria that implement certain principles developed for rulebase evaluation. The principles are *Consistency*, *Completeness*, *Irredundancy*, *Connectivity*, and finally, *Distribution*.

Several errors were detected in the delivered rulebase. Some of these errors were easily fixed. Some errors could not be fixed with the available information. A geometric model of the thruster arrangement is needed to show how to correct certain other distribution anomalies that are in fact errors.

The investigations reported here were partially supported by The Aerospace Corporation's Sponsored Research Program. The author would like to thank the members of the Vehicles Project at Aerospace for a continual stream of hard questions, and Chris Culbert of NASA JSC for providing the rulebase and the challenge to analyze it.

1 Introduction

This paper describes the analysis of an application rulebase for fault diagnosis. The rulebase describes fault detection procedures, experimental procedures to isolate the faults to particular components, and corrective actions that can be applied in a few cases.

The rulebase analysis algorithms are derived from mathematical and computational criteria that implement certain "correctness" principles developed for rulebase evaluation. The principles are *Consistency*, *Completeness*, *Irredundancy*, *Connectivity*, and finally, *Distribution*. Several errors were detected in the delivered rulebase.

An alternative to the systematic analyses above is a model-based validation, which uses several explicit models of system behavior to analyze the behavior of the rulebase that purports to describe the system. This technique is complementary to the systematic criteria, and tends to find different kinds of errors. In fact, each different style of analysis finds somewhat different errors, and it is the recommendation of this paper that many different V&V analyses be performed on any critical rulebase. A geometric model of the thruster arrangement could be used to show how to correct certain other distribution anomalies that are in fact errors.

1.1 Manned-Maneuvering Unit

The Manned Maneuvering Unit (MMU) is essentially a backpack unit for moving a human astronaut around a spacecraft in space. In order to provide maneuverability, there are several thrusters oriented in various directions, and Hand Control Devices for useful groups of them. The thrusters use Nitrogen Dioxide (NO_2) gas for motion.

The FDIR rulebase (see [Lawler,Williams]) is concerned with the problem of fault diagnosis, isolation, and recovery (FDIR) for the MMU. Its purpose is to determine whether the MMU has a fault, to isolate the fault to a particular

subsystem when possible, and to take corrective action when that is possible. The rulebase has 104 rules, written in the expert system shell CLIPS (see [Culbert]), the C Language Interface to Production Systems, developed at NASA's Johnson Space Center. No external functions are called (CLIPS allows externally provided functions to be invoked during hypothesis examination and conclusion generation), so the CLIPS code is self-contained. The MMU FDIR rulebase was kindly provided to us by Chris Culbert of NASA JSC, as was CLIPS.

The rulebase was analyzed according to many of the criteria discussed in the next section. There was no automatic version of any of the analyses, since the criteria are not yet implemented in programs. The criteria were applied by hand, using editors, pattern searching programs, and other text manipulation programs generally available under UNIX. For this rulebase, some extra semantic information is available, such as the symmetry between side a and side b. This information was very useful in the analyses.

2 Principles of Rulebase Correctness

This section describes the correctness principles used for the analysis (see [Landauer89], [Landauer90] for more discussion). The five principles are accompanied by mathematical and computational criteria that serve as specifications of analysis algorithms for rulebases. The *Consistency* criteria address the logical consistency of the rules, and can rightly be considered as "correctness" criteria. The *Completeness* and *Irredundancy* criteria preclude oversights in specifications and redundancy in the rules, and are more like "reasonability" criteria for the terms in the rules. The *Connectivity* criteria concern the inference system defined by the rules, and are like completeness and irredundancy criteria for the inference system (see [Bellman,Walter], [Bellman] for arguments that redundancy in rulebases is dangerous, not just wasteful). Finally, the *Distribution* criteria are "esthetic" criteria for the simplicity of the rules and the distinctions they cause, and the distribution of the rules and the values implied by them.

The approach taken in this section is to treat rulebases as mathematical objects and develop criteria for acceptability, both correctness criteria and "reasonableness" criteria. The criteria should identify inconsistent or awkward rule combinations.

2.1 Rulebase Definitions

A rulebase is a finite set R of pairs

$$r = (hyp, conc)$$

of assertions (or formulas), to be interpreted as

if hypothesis *hyp*, then conclusion *conc*.

The first component (the hypothesis) of a rule r is written $hyp(r)$ and the second one (the conclusion) as $conc(r)$ when there is need to refer to them separately. Each of these parts is considered to be a Boolean function.

The set V of variables in a rulebase R is finite. A *situation* is an instantiation of all of the variables, with the further restriction that all the rules are true of all situations. Every variable is considered to be a feature of the situation, with a possibly unknown value in the appropriate domain. The rest of this section will explain what this restriction means.

Each variable v is considered to be a function applied to situations, so for a situation s, the expression $v(s)$ denotes the value of the variable v in situation s. More generally, for any expression e over a set W of variables contained in V, $e(s)$ denotes the value of the expression in situation s.

The set of situations is therefore a subset of the Cartesian product of all of the variable domains, but the particular subset is not precisely known, since it is limited by the rulebase to only those elements of the Cartesian product that satisfy the rules (i.e., the rules define the situations). There are connections between variables that allow some of them to be computed from others. The Cartesian product will occasionally be called the *situation space*, to distinguish it from the set of situations. An element of the situation space may be called a *prospective* situation until it is determined whether it is actually a situation or not. So the syntactic restriction of having each variable value in the appropriate domain suffices to define the prospective situations, and the semantic restriction that all rules are satisfied defines those prospective situations that are situations.

A rulebase is applied to a situation to compute some variable values (not to set the values, but to find out what the values are), so that a situation has both provided variable values ("input" variables) and derived variable values, some of which are displayed ("output" variables). It is further assumed that the variable values not specified by the input are defined but unknown, and that the rulebase is expected to compute the output variable values.

Rules are implicitly universally quantified over situations. A variable v in the rulebase is a fixed component selection function v applied to a variable situation s. There are no explicit quantifiers, so all situation variables are free in the expressions.

2.2 Analysis Tools

This section describes several derived combinatorial objects and other analytical tools that are useful for analyzing a rulebase. They are primarily graph theoretical notions, including graphs and incidence matrices.

2.2.1 Incidence Matrices

The simplest incidence matrix of a rulebase is called simply *the incidence matrix* of the rulebase. It is indexed by $R \times V$, with entry 1 when variable v occurs in rule r (the occurrences must be free, which is easy now when there are no quantifiers).

It is often convenient to retain the number of occurrences of variables in rules. The *counting* incidence matrix RV of a rulebase is a matrix indexed by $R \times V$, with

$$RV(r, v) \quad = \quad \text{number of occurrences of variable } v \text{ in rule } r,$$

so it may have counts greater than one.

The only non-trivial operation that can be performed on this matrix is multiplication. Since there is only one matrix at present, it must be multiplied by itself. Since the coordinate index sets are not the same, either one of the matrix factors must be transposed (giving actually two different products). The only remaining question is what the products might mean. It turns out to be relatively easy to interpret both of them.

With this matrix RV, the (v, w) entry of the product, $(RV^{tr}\ RV)(v, w)$, is the number of pairs of instances of variable v and variable w contained in the same rule, and the (q, r) entry of the product, $(RV\ RV^{tr})(q, r)$, is the number of pairs of instances of rule q and rule r containing the same variable.

The two matrix products above give rise to two undirected graphs, the first one with variables as vertices, and edges for nonzero entries in the product $(RV^{tr}\ RV)$, and the second with rules as vertices, and edges for nonzero entries in the product $(RV\ RV^{tr})$. The first graph connects two variables if they appear together in a rule, and the second one connects two rules if they have common variables. More detailed graphs will be studied later on, but all will use the same basic construction.

There are several other incidence matrices that are useful for rule analyses, including a clause-variable incidence matrix CV, and a rule-clause incidence matrix RC, but they are analogous to the rule-variable incidence matrix RV and are not described in detail. For this purpose, a clause can be considered as a predicate expression, and C is the set of clauses.

2.2.2 Clause Graphs

The inference C graph has vertices for all clauses c, and an edge from clause c to clause d whenever there is a rule r with $c \in hyp(r)$ and $d \in conc(r)$. The inference R graph has vertices for all rules r, and an edge from rule q to rule r whenever there is a clause c which is in both $hyp(r)$ and $conc(q)$. These graphs are defined from the counting incidence matrices to have labels according to the appropriate counts.

2.2.3 Association Matrices

An association matrix is a covariance matrix computed from occurrence patterns across a set of possible locations. The counting incidence matrix product $(RV)(RV^{tr})$ counts variables in common to rules, measuring the occurrence

pattern of a rule according to the variables it contains. Then the correlations can be computed from the covariances, in the usual way:

$$\begin{aligned} Corr(q,r) &= Covar(q,r)/(Stdev(q) * Stdev(r)), \\ Stdev(q) &= \sqrt{Covar(q,q)}, \\ Covar(q,r) &= (RV\ RV^{tr})(q,r)/|V| - Avg(q) * Avg(r), \\ Avg(q) &= \sum(\text{variables } v \in V)\ RV(q,v)/|V|. \end{aligned}$$

Here, the q row of the counting incidence matrix RV is the occurrence pattern for rule q, so $Avg(q)$ is the average number of occurrences of each variable in rule q, and $Stdev(q)$ is the standard deviation. There is no random variable here, so there is no point in using the "sample standard deviation". The correlation is a measure of similarity between rules, as measured by the variables in them. The correlation value is 1 if and only if the two rules use exactly the same variables with the same frequency of occurrence of each variable. It will be negative, for example, when the two rules use disjoint sets of variables, and -1 in rare cases only (not likely in a rulebase).

Similarly, the counting incidence matrix product $(RV^{tr})(RV)$ counts rules in common to variables, measuring the occurrence pattern of a variable according to the rules containing it. Correlations are computed as before. Other incidence matrices for variables in clauses and clauses in rules can also be used in this way.

The use of correlations is in detecting unusual ones. If clause b almost always occurs with c, then something should be noted when they do not occur together. If variable v always occurs with w, then there may be a good reason for combining the variables. There should also be some justification for unusual correlations or distinctions.

Two rules that use the same variables are not necessarily redundant. As an artifact of the balance criteria described later, it will often be the case that there are sets of rules all using the same variables, giving the rulebase a natural clustering into groups.

Since each covariance matrix above is symmetric and positive semi-definite (as are the corresponding correlation matrices), one can consider computing eigenvectors to determine an "information space", as is done in associative information retrieval systems (see [Landauer,Mah]). The general idea begins with an arbitrary rectangular matrix B, indexed by $R \times C$ (these indexes are just rows and columns for this discussion; any of the incidence matrices or their transposes can be considered). First the association matrix A (indexed by $R \times R$) is computed as the transpose product $(B\ B^{tr})$, then the eigenvectors of the resulting matrix A are found. The eigenvector computation is not too hard, since A is symmetric and positive semi-definite.

This process of determining an abstract space in which to interpret some kind of measurement data is a special case of Multidimensional Scaling, and the eigenvector computations are the same mathematical procedures used in factor analysis and principal components analysis in statistics and pattern recognition (see [Gnanadesikan]).

It often turns out that the number of dimensions is too large to make eigenvector computation desirable. In those cases, the similarity measurements contained in the correlation matrix can be used in a cluster analysis. Clusters are cheap eigenvectors, and most simple clustering methods can give useful information (see [Sibson]). If the rows of B are considered as vectors in an information space, then the clusters of rows are sets of row items using related information.

Correlations can be used to check for some variable or expression dependencies, and particularly, almost dependencies (if a variable v almost always depends on a variable w, then something should be noted when it does not). If two expressions are highly correlated, then their values are almost related by a linear expression. The converse is also true, but correlations do not help directly with non-linear (i.e., almost all) relationships. However, if arbitrary functional transformations of the expressions can be made before the correlations are computed, then the correlations will help again. The problem becomes one of finding out whether or not there is a functional relationship, and finding its form (at least approximately) if there is one. This process is related to dimensionality reduction methods, such as nonlinear scaling or projection pursuit (see [Gnanadesikan], [Huber]), and is an important model construction method.

2.3 Criteria for Rulebase Correctness

This section describes some principles of rulebase correctness, and ways to test them for a particular rulebase. There is no description of how to determine whether or not to test the principles, since that decision is rulebase dependent. A principle of rulebase correctness is a condition on a set R of rules that is required for the rulebase to be reasonable in some incompletely defined sense. This notion is not the same as a principle of modeling a process or a system by rules (that step is hard). It is a notion of how rules fit together into a rulebase.

The five principles so far identified are:

- Consistency (no conflict),
- Completeness (no oversight),
- Irredundancy (no superfluity),
- Connectivity (no isolation), and
- Distribution (no unevenness).

These principles are implemented by many criteria for rulebase correctness. The criteria are separated into classes, according to the principles they implement. The criteria address logical consistency of the rules, completeness of specification of the rules, redundancy of the rules, connectivity of the rule and inference system, simplicity of the rules and the distinctions they cause, and the distribution of the rules and the values implied by them.

The first three principles, Consistency, Completeness, and Irredundancy, are not discussed in detail in this paper, since they are relatively easy to explain (see [Landauer89], [Landauer90] for the full discussion). The Connectivity and Distribution principles are discussed in detail in the next sections.

The Consistency principle leads to criteria that involve some kind of lack of conflict among rules. The idea is that the situations should be well defined, as should all the interesting variable values. The criteria will not be listed here, as they correspond to easy syntactic checks.

The Completeness principle leads to criteria that involve some kind of universal applicability of the rulebase. Defaults are usually used to guarantee certain kinds of completeness. All detectable places where defaults will be used should be signaled, since some of them may only indicate undesired incompleteness in a rulebase, instead of one expected to be fixed by the use of defaults. These criteria will also not be listed here.

The Irredundancy principle leads to criteria that insist that everything in the rulebase is there for some good reason. The variables make a difference, the rules make a difference, and there are no extraneous variables or rules.

2.3.1 Connectivity Criteria

These criteria collect rules together, involving either the entire dynamic process of inference, or the resulting graphs.

Criterion: recursion is dangerous

> The inference R graph should have no cycles.

Similarly for the inference C graph.

Dangling hypotheses and conclusions can be found very easily by looking for vertices in the clause graph (the inference C graph) that have no out-edges or no in-edges.

The rest of the criteria require the deduction graph to be nice in some sense. Disconnected components of the graph have no interaction, so they can be analyzed separately. There is some evidence to the effect that they should be described in different rulebases, instead of combining all the rules into one rulebase.

It is easy (though not necessarily fast) to check a finite directed graph for connectivity and for cycles.

The inference C graph has vertices for all clauses $c \in C$, and an edge from clause c to clause d whenever there is a rule r with $c \in hyp(r)$ and $d \in conc(r)$. A vertex with no out-edges is a clause c with no rule r having $c \in hyp(r)$ and $conc(r) \neq \emptyset$ (so c should involve only output variable values, or else it should not be in $conc(q)$ for any rule q, so that no inference chain can conclude that c holds). A vertex with no in-edges is a clause c with no rule r having $c \in conc(r)$ and $hyp(r) \neq \emptyset$ (so c should involve only input variable values, or else it should not be in $hyp(q)$ for any rule q, so that no inference chain can require that c holds).

The inference R graph has vertices for all rules $r \in R$, and an edge from rule q to rule r whenever there is a clause c which is in both $hyp(r)$ and $conc(q)$. A vertex with no out-edges is a rule r with no clause in $conc(r)$ and in $hyp(q)$

for any rule q (so any clause in $conc(r)$ should only involve output variables). A vertex with no in-edges is a rule r with no clause in $hyp(r)$ and in $conc(q)$ for any rule q (so any clause in $hyp(r)$ should only involve input variables).

2.3.2 Distribution and Simplicity Criteria

This section describes some of the simplicity and distribution criteria that can be used to signal possible problems with a rulebase. All of the criteria involve the way the rules divide up the set of situations. None of them is a mathematical correctness criterion; only a kind of "esthetic" criterion.

Criterion: simple distinctions

> For every rule r,
> the set of situations satisfying $hyp(r)$ is simple.

Each rule r provides a distinction in the set S of situations between those situations s for which r acts and those for which it passes. When the boundary between those sets is too complicated, the expressions used in the hypothesis of r are awkward (and vice versa). It is sometimes necessary to use awkward phrases or distinctions in the rules, but some justification should be provided. Note that some awkwardness can be removed by using more than one rule in some cases.

Criterion: compact variable distribution

> For every variable v,
> the set of rules accessing v should be a small part of the entire rulebase.

This criterion affords a kind of modularity. The references to any one variable should be well-localized. A weaker form of the criterion would only require localization for the variables that occur in rule hypotheses. In any case, some variables (such as system health) must occur in many or all rules, but their wide distribution should be justified.

The other criteria describe various distributions as even. In this context, "even distribution" is less stringent than uniform distribution, and it really only means "not very non-uniform"; it represents a kind of balance condition. Cases of uneven distribution should be justified. It is clear that rulebases containing rare special cases will not satisfy these criteria. Part of the purpose of these criteria is to call such cases to the attention of the rulebase designer. The situations satisfying a given rule hypothesis should be evenly distributed in the variable domains. The rules accessing a given variable should be evenly distributed among its possible values.

Finally, The set of rules should be evenly distributed among variables. This criterion would prevent a larger number of rules from accessing (or just reading) one variable than for another. During rulebase development, some aspects of situations are not fully implemented in the rules, so some variables have very few references. This criterion signals those variables for further work (or justification).

The most blatantly non-uniform distributions are caused by unusual special cases. For example, if two variables always occur together except in one rule, or if two variable values are always correlated except in one rule, then the exception is an anomaly. In either case, some justification is required, either that there is a real difference for that one rule, or that there is a reason to have two variables where one might suffice.

The criterion examines the distribution of the rules over V. For a given variable v, the number of rules that access v is the column sum in column v of the counting incidence matrix RV. The row sum of row r of RV counts the number of variables mentioned in rule r. This count is related to the simplicity of $hyp(r)$.

2.3.3 Distribution Checking

Distribution checking is not a well-established analysis technique. This section describes a test for each of the distribution and simplicity criteria defined earlier.

Using prospective situations, simple distinctions means only that $hyp(r)$ is simple in form. Without using the entire rulebase to determine the set of situations, this is about the only thing one can check along these lines.

Compact variable distribution is easier to check. The column sums of the incidence matrix RV count how many rules contain the column variable v. Then v has a compact distribution if the sums are small. Uniform variable distribution

also uses column sums of RV, checking that the numbers for a given variable v are all about the same (it should be noted that these criteria are more or less opposing, in that one wants all the values small and the other wants most of the values zero).

The uses of association and correlation matrices are even less well established. The basic idea comes down to one question (expressed here only for variables, but equally applicable to clauses or rules or other constructions):

> If v and w are highly correlated,
> then why are they different?

Detecting unusual conditions requires some computational indication of what the usual conditions are. For any given computational definition of the usual conditions, the cases not satisfying it can be determined (it is only after some empirical examination that the usual conditions can be computed, and deviations from that can be deemed unusual). For example, if variables v and w almost always determine expression e, then the usual case has the value of e for a particular situation dependent on the value of the pair (v, w) for that situation. Then there is some function of the pair (v, w) that should be nearly the same as the value of e, almost all the time. With the assumptions, it is now sufficient to distinguish large differences from small ones. Of course, there is still the problem of distinguishing small fluctuations (changes that do not indicate a new trend) in the usual values from the first signs of a real change in the usual values.

2.3.4 Other Criteria

This section contains some analyses that should lead to some other criteria, though more work is needed on each of them.

The various uses of correlation matrices to analyze rulebases are not so well established that they can be elevated to rulebase criteria. The simplest analysis considers only the pairs of items that have very high correlations (close to 1 or -1). Highly correlated variables, clauses, and rules might benefit from being rearranged to reflect the information structure better. For example, if two variables are highly correlated (over their sets of instances in clauses or rules), it might be better to express them both as deviations of some kind from a common variable. For another example, if two clauses have a correlation of -1, then they occur in large disjoint sets of rules (or they contain disjoint sets of variables, depending on which correlation matrix is used), so they are nearly mutual negations, and it might be better to replace one of them with the negation of the other. In this case (and, indeed, in all cases of high or unusual correlations), the correlation information is a derived feature of the rulebase, and may explain some facet of the system being modeled that was not previously seen as significant (or even noticed). It might therefore be better to leave the rulebase as it is until a sufficient explanation is found.

The association matrices to be considered are computed from the rule-clause incidence matrix RC, the clause-variable incidence matrix CV, and the rule-variable incidence matrix $RV = RC * CV$.

The main intent of these considerations is to find goodness criteria that can be evaluated using these association matrices. Until such time as they can be properly formulated, however, there are still some interesting questions. For example, what does it mean for all the eigenvalues to be the same size? What does it mean for one eigenvalue to be much larger than the rest? The hardest problem is not computational, but interpretational: to explain the dimensions in the information space (the principal components). There is still some controversy in whether or not there is any meaning in these inferred axes, even though (or perhaps because) the technique has been used in statistical analyses for many years.

3 MMU Analyses

Many analyses were performed that are implementations of the criteria discussed in the previous sections. There was no automatic version of any of the analyses, since the criteria are not yet implemented. The criteria were applied by hand, using editors, pattern searching programs, and other text manipulation programs available under UNIX. For this rulebase, some extra semantic information is available, such as the symmetry between sides a and b. This information was very useful in the analyses.

1	thruster
40	thrusters

Figure 1: Some Term Frequencies

1	no-xfeed-fuel-reading-test-side-a-grt
1	no-xfeed-fuel-reading-test-side-a-lss
2	no-xfeed-fuel-reading-test-side-b-grt

Figure 2: Some Rulename Frequencies

3.1 Preliminary Analyses: Uninterpreted CLIPS

The first analysis used a simple editor script, with (almost) no CLIPS knowledge beyond what can be found by looking at a CLIPS rulebase (which looks vaguely like LISP, with terms grouped together using parentheses). From the original rulebase file, all strings were mapped to "..." to avoid any reliance on word meanings. The names were selected from the text (here is where the CLIPS knowledge was used, in that the minus sign "-" can be part of a name instead of a delimiter). The left parenthesis "(" was also kept to separate function names from other names. Then the names were extracted, sorted, and counted to make a reference file.

This simple form of analysis found the first two errors. Among the name frequencies are the lines in Figure 1. The isolated instance is a mistake. In rule "xfeed-fuel-reading-test-general", there is a clause error. The delivered rulebase has

(checking thruster),

but it should have

(checking thrusters)

instead.

The second error is an incorrect rule name. The second instance of rule name "no-xfeed-fuel-reading-test-side-b-grt" is wrong. It should be "no-xfeed-fuel-reading-test-side-b-lss". This error was found as a side (a,b) asymmetry in the name frequencies shown in Figure 2.

Another anomaly found in the frequencies is not an error. The frequencies shown in Figure 3 might indicate an inconsistent use of the terms "cea-a-b" and "cea-coupled" that should be the same term. However, the two terms do mean something different in the rulebase, so the anomaly is not an error.

3.2 Detailed Analyses: Partially Interpreted CLIPS

A more systematic analysis based on the existing criteria was also conducted by hand. It differed from the preliminary analysis primarily in the degree of knowledge of CLIPS that was used in the editing process. Using this knowledge is equivalent to interpreting some of the symbols found in the rulebase, for example, in order to distinguish CLIPS commands from MMU terms.

4	cea-a
2	cea-a-b
4	cea-b
2	cea-coupled

Figure 3: More Term Frequencies

```
1    (vda a b1 off) (vda a b4 off) (vda a ?n&~ b1&~ b2&~ b3&~ b4 on)
1    (vda a b1 off) (vda a b4 off) (vda a ?n&~ b1&~ b4 on)
1    (vda a b1 off) (vda a f2 off) (vda a ?n&~ b1&~ f2 on)
4    (vda a b1 off) (vda a f3 off) (vda a ?n&~ b1&~ f3 on)
```

Figure 4: Some Lines from "vda" clause file

Some rulebase properties were found to be useful in this analysis that were not described in the previous section, and had not been considered as criteria for rulebase analyses (see [Landauer89]). The new criteria found during the analysis involve symmetry between side a and side b of the MMU, and, more generally, the symmetry among the replicated thrusters. The question to be asked in this case is, Do the multiple versions of a replicated object occur the same number of times in the rulebase, and if not, then why not? These criteria are associated with the distribution principle, and they simply say that any problem symmetries should be reflected in the rulebase, so that they appear in the distribution summaries.

Another kind of symmetry question, which not only concerns replicated objects, asks how to use geometric models in a "good" way. For the MMU, the thruster geometry is important in checking that the combinations of thrusters specified by the rules for correcting attitude and position errors correspond properly to the motions required to correct those errors. Because this geometric model was not provided with the rulebase, that analysis was not done.

3.2.1 Analysis Preparation

This analysis began with a revised rulebase, in which the two errors found earlier were corrected. They are clearly syntactic errors, and were fixed without further analysis. The new rulebase file was edited, using a knowledge of CLIPS syntax to identify terms and clauses, and to separate hypotheses of rules from conclusions. Many different syntactic items were separated: rule names, strings, functions, terms, and clauses were all placed into separate files of code numbers (to remove traces of semantic information derivable from the names). Editor scripts were made to translate items to code numbers, then several versions of the rulebase were made by partial translation.

The "separated rule file" was made by combining all hypotheses in each rule into one line, and all conclusions in each rule into one line. A file was made from the separated rule file to show clauses appearing in rule hypotheses, and a large number of different "(or " clauses was noted. files were made to show clauses appearing in rule conclusions, (except "(printout " clauses, which were omitted from the analysis, since the proper spelling and explanations for detected faults were not part of this analysis), and clauses appearing in each rule. These were used for the matrix and graph analyses.

3.2.2 Amplifiers

By far the biggest number of different clauses occurs for the clauses having function "(vda ", which concern the valve drive amplifiers (VDAs), each of which is used to control a thruster. Many of these clauses are collected in triples with "(or ". A file was made that contains the "(or " expressions with "(vda " clauses. Some lines of the file are shown in Figure 4 (the first column is a frequency count). The question marks in these clauses indicate variable names. For example, in the last line above, the third clause means that some side a thruster other than "b1" or "f3" is on.

The thruster names were selected as thruster names by context, manually. Every one occurs in a "(vda " clause, and it appears that every name that occurs as the third entry in a "(vda " clause is a thruster name. The spellings of the thruster names determine the original grouping, as shown in Figure 5.

The arrangement of thrusters and their relationship to roll, pitch, and yaw, and to rotation and translation will have to come from a geometric model of their locations and directions. Such a model is necessary for validation of the thruster commands.

Examining the "vda" clause file leads to the first thruster anomaly. Some clauses have all four "f" or "b" thruster names, and some do not. It turns out that the clauses with all four thruster names also have both sides on, and the clauses without all four have only one side on if the same thruster group is used (e.g., both "f" or both "b"), and both sides if different thruster groups are used. Since a model of the thrusters was not available, this anomaly cannot be resolved (an anomaly is not necessarily an error, remember, just something strange in the rulebase).

b1, b2, b3, b4
f1, f2, f3, f4
l1, l3
r2, r4
u3, u4
d1, d2

Figure 5: Thruster Name Grouping

b1, b4
b2, b3
f1, f4
f2, f3

Figure 6: Thruster Name Co-Occurrences

A file was made from the "vda" clause file to show which thrusters in the above groups are associated with each other in the same "(or " combination of "(vda " clauses (the same line in the "vda" clause file). The "f" and "b" groups subdivide, as shown in Figure 6. For example, "b1" and "b2" do not occur together in an assertion unless it asserts that some thruster different from both is on in the third disjunct of an "(or " combination.

A file was made from the "vda" clause file to show which thrusters can be associated with which sides (the side is the second "(vda " clause entry, and the thruster name is the third). This association leads to side assignments for the thruster subgroups above, as shown in Figure 7. The "l", "r", "u", and "d" thrusters can appear with either side, but the "f" and "b" ones cannot (e.g., "b2" never appears with side a). Each of these files was also checked for side a, b symmetry, and no anomalies were found.

The MMU FDIR report says (only indirectly) that there are 24 thrusters, which was originally interpreted to mean that there are six places (the labels "b", "f", "l", "r", "u", "d" are interpreted to mean "back", "front', "left", "right", "up", "down"), with four thrusters in each place; however, not all names occur in the rulebase, so there is a possible symmetry error in allowing "l1" and "l3" for both side a and side b instead of just for one of them, with "l2" and "l4" for the other. Similarly, "r1", "r3", "u1", "u2", "d3", "d4" do not appear, and yet are probably required to make 24 thruster names in all.

3.2.3 Hand Controllers and other Clause Notes

Another large group of clauses are the "(rhc " and "(thc " clauses, which deal with rotational and translational hand controllers. A file was made from the separated rule file to contain all those clauses. The complete file is shown in Figure 8 (the numbers on the left are frequencies). It turns out that every "(rhc " clause is paired with a "(thc " clause, and vice versa (this property was found by observation, but it could have been found by examining the correlations between occurrence patterns of these clauses). These counts also demonstrate the symmetry among roll, pitch, and yaw on the one hand, and x, y, and z on the other.

There are two styles of motion: rotation and translation. The rotations can be roll, pitch, or yaw, representing (it is assumed) the usual notions of vehicle attitude. The translations can be x, y, or z, representing (it is assumed) some

side a
b1, b4
f2, f3

side b
b2, b3
f1, f4

Figure 7: Thruster Name-Side Associations

4	(rhc roll neg pitch none yaw none) (thc x none y none z none)
4	(rhc roll none pitch neg yaw none) (thc x none y none z none)
4	(rhc roll none pitch none yaw neg) (thc x none y none z none)
4	(rhc roll none pitch none yaw none) (thc x neg y none z none)
4	(rhc roll none pitch none yaw none) (thc x none y neg z none)
4	(rhc roll none pitch none yaw none) (thc x none y none z neg)
25	(rhc roll none pitch none yaw none) (thc x none y none z none)
4	(rhc roll none pitch none yaw none) (thc x none y none z pos)
4	(rhc roll none pitch none yaw none) (thc x none y pos z none)
4	(rhc roll none pitch none yaw none) (thc x pos y none z none)
4	(rhc roll none pitch none yaw pos) (thc x none y none z none)
4	(rhc roll none pitch pos yaw none) (thc x none y none z none)
4	(rhc roll pos pitch none yaw none) (thc x none y none z none)

Figure 8: Hand Controller Clauses

24	(side a off) (side b on)
24	(side a on) (side b off)
43	(side a on) (side b on)

Figure 9: Side Clause Combinations

unspecified Cartesian coordinate system. The hypotheses of a single rule have changes in at most one component of at most one style of motion, and the changes occur symmetrically among the components. The relationship between the component being corrected and the combination of thrusters used to correct it cannot be checked, because no geometric model is available. An internally consistent relationship could be derived from the rules, but would not necessarily be correct.

Another large group of clauses is the "(side " clauses; a "side" clause file was made to contain them. They always occur in pairs, one for side a and one for side b, The pairs are shown in Figure 9. The few rules that do not have these clauses in their hypotheses are mostly in the rulebase to control other groups of rules, or to print out the problem statements (the rulebase has five predefined scenarios; special rules print the corresponding problems and solutions). There are no clause combinations for the case in which side a and side b are both off.

Other coverage notes were found by examining other kinds of symmetry. The "(aah " clauses, involving the Automatic Attitude Hold (AAH) process, and the "(gyro " clauses, involving the gyroscopes, form another potential source of error, since they split the attitude control information in two ways. The statistics of these clauses and their cooccurrences were computed, and are shown in Figure 10 and Figure 11. First, some simple anomalies are obvious at this point. There is no clause "(gyro off)" in any rule. There is no applicable rule if "(gyro off) and (aah on)".

Finally, another anomaly that is certainly an error was found by trying to infer from the above tables how these clauses combine in threes. It can be explained more easily, however, by noting that there is no combination of clauses "(aah on)" and "(gyro movement roll pos)". In fact, examining the four rules containing "(gyro movement roll pos)" shows

53	(aah off)
20	(aah on)
49	(gyro movement none none)
4	(gyro movement pitch neg)
4	(gyro movement pitch pos)
4	(gyro movement roll neg)
4	(gyro movement roll pos)
4	(gyro movement yaw neg)
4	(gyro movement yaw pos)
73	(gyro on)

Figure 10: Clause Counts for Attitude Clauses

49	(aah off) (gyro movement none none)
4	(aah off) (gyro movement roll pos)
53	(aah off) (gyro on)
4	(aah on) (gyro movement pitch neg)
4	(aah on) (gyro movement pitch pos)
4	(aah on) (gyro movement roll neg)
4	(aah on) (gyro movement yaw neg)
4	(aah on) (gyro movement yaw pos)
20	(aah on) (gyro on)
49	(gyro movement none none) (gyro on)
4	(gyro movement pitch neg) (gyro on)
4	(gyro movement pitch pos) (gyro on)
4	(gyro movement roll neg) (gyro on)
4	(gyro movement roll pos) (gyro on)
4	(gyro movement yaw neg) (gyro on)
4	(gyro movement yaw pos) (gyro on)

Figure 11: Pair Counts for Attitude Clauses

that the clause "(aah off)" is used instead. The same error also appears in the count for "(aah on) (gyro on)", which is 20 instead of 24, and in the count for "(aah off) (gyro on)", which is 53 instead of 49. The numbers 49 and 24 are much more consistent with the hand controller counts than 53 and 20 are.

3.2.4 Clause and Rule Associations

In order to compute associations, several files were made for incidence matrices and counts. The incidence matrix for clauses vs. rule hypotheses is sparse, with an entry for each rule that consists of a list of the clauses in the rule's hypothesis. The clause count vector has an entry for each clause that contains the number of occurrences of the clause in rule hypotheses. The co-occurrence matrix for clauses is also sparse, with an entry for each pair of clauses that occur together in a rule hypothesis. The entry is the number of rule hypotheses in which the two clauses occur together. The clause pair count vector has an entry for each clause that counts the number of clause pairs in which it occurs. These two count vectors are different, with the second one always larger. If a clause c occurs in exactly one rule hypothesis h, then the corresponding entry in the clause count vector will be one, and if that rule hypothesis h has four clauses, then the entry in the clause pair count vector for c will be three (one for each of the other clauses in the rule hypothesis).

Files were made for the clause count vector, the clause co-occurrence matrix, and the clause pair count vector.

The data files were converted, by editing them systematically, into two programs to compute correlations. For each clause, the first program ("frac.c") computes and prints the fraction of its co-occurrences with each other clause (when they do occur together). For each pair of co-occurring clauses, the second program ("corr.c") prints the correlations.

Suppose that each clause c has frequency $f(c,r) = 0$ or 1 in each rule r. Suppose also that there are nr rules and nc clauses. The rules are considered as samples, so each clause is considered as having some kind of clause distribution over the rules (not necessarily a random distribution), and various statistical measures can be computed. The clause count for clause c is $s(c) = \sum_r f(c,r)$, so its average frequency across the rules (the fraction of rules it is in) is $avg(c) = s(c)/nr$ and its variance is

$$var(c) = \frac{s(c)(nr - s(c))}{nr^2}$$

(since $f(r,c)^2 = f(r,c)$). The clause pair frequency for clauses c and d is

$$m(c,d) = \sum_r f(c,r)f(d,r)$$

and the clause pair count is

$$p(c) = \sum_{d \neq c} m(c,d)$$

$$= \sum_r f(c,r) \sum_{d \neq c} f(d,r)$$
$$= \sum_r f(c,r)(h(r)-1)$$

for all clauses c, where $h(r)$ is the number of clauses in rule r. The correlation is computed in the usual way:

$$corr(c,d) = \frac{m(c,d)/nr - avg(c)\ avg(d)}{\sqrt{var(c)}\ \sqrt{var(d)}}$$

for all clauses c, d, which can be simplified to

$$sv(c) = \sqrt{s(c)\ (nr - s(c))}$$
$$corr(c,d) = \frac{m(c,d)\ nr - s(c)\ s(d)}{sv(c)\ sv(d)}$$

for all clauses c, d.

The program "frac.c" is made from the clause co-occurrence matrix file and the clause pair count file to print co-occurrence fractions. Lines of the form

```
c000 9
```

in the clause pair count file become lines of the form

```
double c000 = 9;
```

in the program "frac.c". Lines of the form

```
1 c000 c022
```

in the clause co-occurrence file become lines of the form

```
printf("c000,c022 = %.4f\n",1/c000);
```

in the program "frac.c". The program then simply prints out the computed fractions.

The program "corr.c" is made from the clause co-occurrence matrix file and the clause count vector file to print correlations. Lines of the form

```
1 c000
```

in the clause count vector file become lines of the form

```
double c000 = 1.0, stdc000 = sqrt(1.0 * (nr - 1.0));
```

in the program "corr.c". Lines of the form

```
1 c000 c022
```

in the clause co-occurrence matrix file become lines of the form

```
printf("c000,c022 = %.4f\n",(1*nr-c000*c022)/(stdc000*stdc022));
```

in the program "corr.c". The program then simply prints out the computed correlations.

Then a file was made that contains all correlations above 0.7, sorted in decreasing order by correlation. Many clause pairs had a correlation of 1.0; almost all of the clauses in those pairs occurred exactly once in the rulebase, a few occurred twice, and one pair occurred six times each, all in the same rules. No anomalies were detected.

The largest correlation less than one is 0.9259, between the clauses "(aah off)" and "(gyro movement none none)", since only 4 of the 53 instances for the former clause do not occur with the latter clause. This is an anomaly, and in fact, it is the same error as the one described above. The next highest correlation is between a clause "(failure ?)", which only occurs in the combination "(not (failure ?))" (meaning that there is no asserted failure), and the two clauses "(xfeed-a closed)" and "(xfeed-b closed)" (separately). Only one rule contains the former clause without the latter clauses, which always occur together; these are the two mentioned above that occur in six rules. The extra rule containing the failure clause is a control rule that begins the tank and thruster test (most of the rules concern the electronics and not the propulsion system). This anomaly is not an error.

3.3 Discussion

The inference path analyses were not performed on this rulebase, due to their large computational requirements. It is expected that after the rulebase anomalies are corrected, an inference graph analysis will be performed.

It should be noted that many of the tests did not identify any anomalies. This situation is not a problem; because the theory is to apply as many tests as practical, there will often be tests that do not find errors. Moreover, in the rare cases of a correct rulebase, none of the tests will find any errors.

Special purpose tests, using special purpose criteria, will always be useful in analyzing a complex rulebase. The important point here is to make the special test usefully special, instead of making it the most general test possible. Some criteria can be made more widely applicable, and some will remain special purpose.

In the case of the MMU analysis, the symmetry criteria can be applied in general to systems with replicated components, but the choice of symmetries for the thrusters is specific to the MMU. The unmodeled geometric relationships among the thrusters was a missing aspect of the MMU definition that would have greatly assisted the analysis. A geometric model would allow validation of the rules that relate the VDA effects with the thrusters that cause them, and the rules that group thrusters together.

Perhaps the most interesting result of this analysis is that the tests that discovered errors did not do so automatically. In some cases, it was not at all obvious that the data represented errors. Some thought about the data and interpretation of results was required. It is not likely that a completely automatic system will find all errors in a rulebase (even aside from the undecideability barrier). A certain care will also be necessary.

However, these analyses and tools to implement them make the process of discovering some kinds of errors much easier, and should thereby make the design process much more effectively free of such errors.

4 References

[Bellman]

Kirstie L. Bellman, "The Modeling Issues Inherent in Testing and Evaluating Knowledge-based Systems", *Expert Systems and Applications*, Pergamon Press (to appear, 1990)

[Bellman,Walter]

Kirstie L. Bellman and Donald O. Walter, "Analyzing and Correcting Knowledge-Based Systems Requires Explicit Models", *Proc. AAAI 1988 Workshop on Verification and Validation of Knowledge-based Systems*, AAAI (1988)

[Culbert]

Chris Culbert, "CLIPS Reference Manual" (Version 4.2), NASA Johnson Space Center (April 1988)

[Gnanadesikan]

R. Gnanadesikan, Methods for Statistical Data Analysis of Multivariate Observations, Wiley (1977)

[Huber]

Peter J. Huber, "Projection Pursuit" (with discussion), *The Annals of Statistics*, Vol. 13 No. 2, pp. 435-529 (1985)

[Landauer89]

Christopher Landauer, "Principles of Rulebase Correctness", in Kirstie L. Bellman (ed.), *Proc. IJCAI 89 Workshop on Verification, Validation, and Testing of Knowledge-Based Systems*, Detroit, Michigan, 19 August 1989, AAAI (to appear, 1990)

[Landauer90]

Christopher Landauer, "Principles of Rulebase Correctness", *Expert Systems and Applications*, Pergamon Press (to appear, 1990)

[Landauer,Mah]

Christopher Landauer, Clinton Mah, "Message Extraction Through Estimation of Relevance", Chapter 8 in R. N. Oddy, S. E. Robertson, C. J. van Rijsbergen, P. Williams (eds.), Information Retrieval Research, "Proc. of the Joint ACM and BCS Symp. on Research and Development in Information Retrieval", Cambridge University, June, 1980, Butterworths, London (1981)

[Lawler,Williams]

Dennis G. Lawler, Linda J. F. Williams, "MMU FDIR Automation Task", Final Report, Contract NAS9-17650, Task Order EC87044, McDonnell-Douglas Astronautics Co. (3 February 1988)

[Sibson]

R. Sibson, "SLINK: An optimally efficient algorithm for the single link cluster method", *Computer Journal*, Vol. 16, pp. 30-34 (1973)