TRANSFORMATION Reborn: A New Generation Expert System for Planning HST Operations

Andrew Gerb
Space Telescope Science Institute[1]
3700 San Martin Dr.
Baltimore, MD 21218

Abstract

The Transformation expert system (TRANS) converts proposals for astronomical observations with the Hubble Space Telescope (HST) into detailed observing plans. It encodes expert knowledge to solve problems faced in planning and commanding HST observations to enable their processing by the Science Operations Ground System (SOGS). Among these problems are determining an acceptable order of executing observations, grouping of observations to enhance efficiency and schedulability, inserting extra observations when necessary, and providing parameters for commanding HST instruments.

TRANS is currently an operational system and plays a critical role in the HST ground system. It was originally designed using forward-chaining provided by the OPS5 expert system language, but has been reimplemented using a procedural knowledge base. This reimplementation was forced by the explosion in the amount of OPS5 code required to specify the increasingly complicated situations requiring expert-level intervention by the TRANS knowledge base. This problem was compounded by the difficulty of avoiding unintended interaction between rules.

To support the TRANS knowledge base, XCL, a small but powerful extension to Common Lisp was implemented. XCL allows a compact syntax for specifying assignments and references to object attributes. XCL also allows the capability to iterate over objects and perform keyed lookup.

The reimplementation of TRANS has greatly diminished the effort needed to maintain and enhance it. As a result of this, its functions have been expanded to include warnings about observations that are difficult or impossible to schedule or command, providing data to aid SPIKE, an intelligent planning system used for HST long-term scheduling, and providing information to the Guide Star Selection System (GSSS) to aid in determination of the long range availability of guide stars.

---

[1] Operated by the Association of Universities for Research in Astronomy for the National Aeronautics and Space Administration

1. Introduction

This paper describes the design and operation of the Transformation expert system (TRANS) which was originally built to interface between the Proposal Entry Processing System (PEPSI) and the Science Operations Ground System (SOGS). When observers submit proposals for use of NASA's Hubble Space Telescope (HST) for astronomical observations, these proposals are entered electronically into PEPSI for syntax checking and rudimentary semantic analysis [Jackson88a]. These proposals need to be processed by (SOGS) which is comprised of the Science Planning and Scheduling System (SPSS), which schedules observations within a week, Science Commanding System (SCS) which generates commanding instructions for an observation, Observation Support System (OSS) which monitors operation of the telescope during the observation and Post Observation Data Processing System (PODPS) which receives and processes science data received from HST. Providing inputs to SOGS requires detailed knowledge that the proposer would not be expected to have. Such knowledge includes the best order in which to perform the observations, how the observations should be grouped to minimize telescope movement and instrument reconfiguration, what extra observations are necessary to support the science requested and attributes necessary to plan, schedule, command and process data for the observations. TRANS was designed to operate on the output of PEPSI, using expert knowledge to provide input to SOGS. The input to SOGS is provided in text form to allow override by an expert.

The original purpose of TRANS has been expanded to allow greater use of its knowledge. It provides input to SPIKE, a knowledge-based long term scheduler for HST observations which determines the optimal assignment of observations to weeks. To aid this process, TRANS also generates requests for the Guide Star Selection System (GSSS) to determine long-range availability of guide stars, needed to insure the pointing accuracy of HST. Trans also generates diagnostic error message to identify observations which will cause trouble for SOGS, SPIKE, GSSS or HST. Figure 1 illustrates the data flow through TRANS.

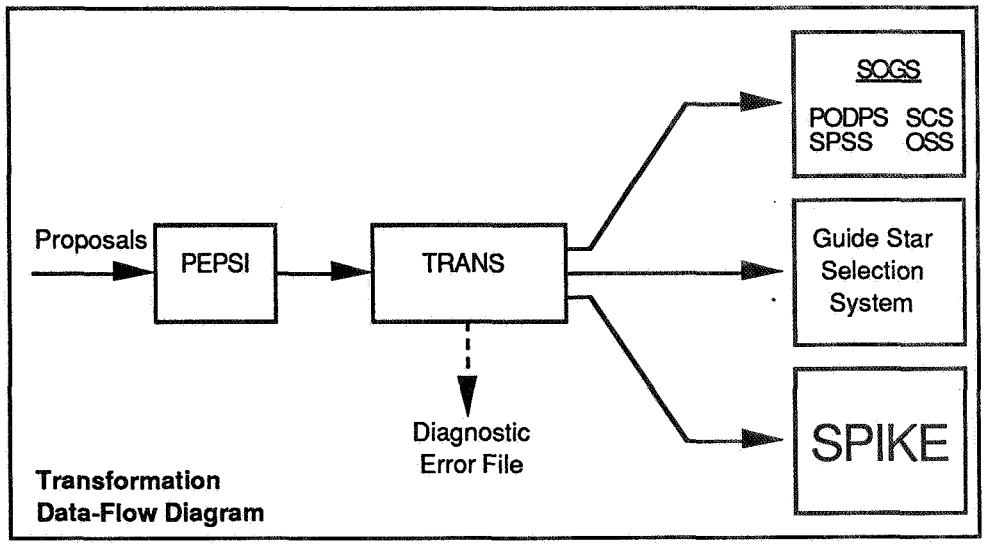

Figure 1

2. The Knowledge Based Functionality of TRANS

Proposers describe their observing program by dividing them up into *exposures* and entering them onto a form called an *exposure logsheet* [Space89a]. The TRANS knowledge base operates on these exposures taken from a database generated by PEPSI based on the exposure logsheet. Trans performs five basic functions on the exposures it receives as input - ordering, merging, inserting extra exposures needed to support those requested, computing attributes and reporting its conclusions [Gerb90a].

2.1 Ordering Exposures

Trans orders exposures based on user-specified constraints. This ordering can be modified (within limits) by SPIKE and SPSS, and is used primarily as a guideline for the grouping of exposures for efficiency purposes. The ordering phase of trans uses technology from the domain of constraint satisfaction problems to insure that all constraints specified by the observer's proposal are satisfied. Details of this process will be made the topic of future publications.

2.2 Merging Exposures

The process of dividing exposures into a hierarchical grouping is called *Merging*. During the process of merging, the ordered exposures are grouped into contiguous disjoint sets called *alignments*. Alignments are then grouped into *observation sets (obsets)* and finally, obsets are grouped into *scheduling units*. These groupings define the data structures used by SPSS to schedule HST observations[Miller87a].

Alignments are the tightest grouping of exposures. All exposures in an alignment must use the same HST pointing and orientation, must generate science or engineering data at the same rate and must have only small gaps or interruptions between successive members. Division of exposures into alignments is important for two reasons. First, exposures in the same alignment can be commanded for much more efficient use of spacecraft time. Second, alignments are the basic units of planning for SPSS; SPSS does not schedule on the exposure level.

Obsets are groups of alignments which can be performed without a change in the operating mode of the pointing control system (PCS). HST usually depends on positional monitoring of pairs of stars (called guide stars) to maintain its pointing. A series of alignments can be in the same obset if they all can use the same guide star pair. Alignments which do not need guide stars, such as those that rely on gyros for their pointing control, can also be grouped into the same obset.

Scheduling Units are groups of obsets that SPSS schedules together. When scheduling an obset requires the next obset to be scheduled immediately afterwards, both are placed in the same Scheduling Unit.

TRANS first merges exposures into alignments. It marches down the exposures in the order determined by the ordering phase. For each exposure, it determines if there is an efficiency reason why it should be placed in an alignment with the previous exposure. If such a reason exists, TRANS then looks for a reason why it should not. If this search fails, the exposure becomes part of the alignment containing the previous. If not, a new alignment starts. This process is repeated for merging alignments into obsets, and again for merging obsets into scheduling units.

2.3 Inserting Extra Exposures

Once exposures are merged, extra exposures or alignments may need to be inserted. There are currently eight cases where this must be done:

1. When the proposer has requested in the proposal that an exposure be performed multiple times, TRANS needs to copy the exposure.

2. When an alignment needs special commanding or requires monitoring of engineering telemetry by a link to the ground or by tape recorder, an extra exposure is added to the beginning of the alignment to account for the time used in this process and to trigger the appropriate commanding.

3. When an alignment uses the Fine Guidance Sensors (FGS) for science purposes, an exposure must be placed at the end of the alignment to trigger commanding to restore the FGS to its original state and to account for the time used in the process.

4. When an obset uses the Faint Object Spectrograph (FOS), an extra alignment with a single exposure is placed at the end of the obset to trigger commanding to restore the FOS to its original state and to account for the time used in the process.

5. When an alignment sends data to the ground in realtime for analysis, an extra alignment with a single exposure is inserted before the alignment which requires the analysis to allow time for the analysis to occur. It is not necessary to insert this alignment if there is already sufficient time to perform the realtime analysis.

6. When an alignment sends data to the ground in realtime to determine the pointing of a later alignment for target acquisition purposes, a special alignment with a single exposure is inserted before the later alignment to trigger commanding for the position uplink and to account for the time used in this process.

7. When an alignment with more than one exposure is found by TRANS to last longer than a reasonable viewing interval for the target requested, TRANS divides its exposures up into several smaller alignments. It marches through the exposures in the alignment, summing the total time. When the time exceeds the maximum time allowed, a new alignment is created containing the remaining exposures. This process is continued until the remaining exposures comprise an alignment short enough to fit in the time allowed. This process could not occur during the merging phase since merging information is required to compute overhead times.

8. If after the process described in step 7, an alignment containing a single exposure is still too long, that exposure is replaced by several shorter copies of itself, each in its own alignment. First an attempt is made to create two exposures, each roughly half as long as the original, and the overhead time is recomputed. If either is still too long, three are created. This process is continued until either the individual exposures are short enough, or TRANS determines that it is not reasonable to create more exposures.

The eighth example above is an especially complex process, as it is not always possible to break exposures evenly. Many of the instruments on HST require the length of their exposures be an integer multiple of some unit of time (often dependent on the parameters with which the exposures are created). It is up to TRANS to determine the apportionment of time among the new exposures to minimize the number of exposures needed to for all exposures to fit.

2.4 Determining Exposure Attributes

Once all exposures, alignments, obsets and scheduling units have been created, there are numerous attributes that need to be computed for each.

2.4.1 Overhead Times

TRANS must compute overhead times for all exposures, alignments, obsets and scheduling units. Computing overhead for exposures involves computing the time involved to set up the observations (opening shutter, moving filter in place, etc.), commanding the observations, taking data, reading the data to tape or to the ground, and restoring the instrument to its original state. For some instruments, it is also necessary to determine whether routine calibrations are needed and how long these will take. To do this, TRANS must consider the length of time since the most recent calibration and opportunities for another calibration later in the alignment.

Computing overhead times for an alignment involves summing the times for each of its exposures. If some are required to be parallel with (occurring at the same time as) others, time must be subtracted to take this into account. If the alignment requires a realtime contact with the ground, extra time must be added. Certain instruments require additional time be added to the beginning or the end of the alignment.

Computing overhead times for obsets and scheduling units involve summing up the times for each of their alignments. TRANS determines the number of times the earth has occulted the targets and adds time to compensate. More time must be added to relocate guide stars at the end of each earth occultation. Whenever the target changes, time must be added to complete the Small Angle Maneuver (SAM) required to point at the new target. Also, at the beginning of each obset, time must be added to locate the guide stars used in the obset.

2.4.2 TDRS Contact Parameters

If a proposal requires the uplink or downlink of data using the NASA Tracking and Data Relay Satellite (TDRS), TRANS must compute the request parameters. These determine whether the link is an uplink or downlink, whether the link involves science data, engineering data, or both, and the data rate and duration of the contact.

2.4.3 Exposure Pointing Information

TRANS records extensive data on the pointing of observations. TRANS decides whether guide stars are necessary, or whether gyros are sufficient to control pointing accuracy. Exposures are marked to be calibration, target acquisition or science exposures. TRANS determines whether the exposure needs to point at a specific target, and if so, whether the pointing can remain the same as the previous exposure. TRANS also computes whether a specific HST roll orientation is required, and exactly where on the HST field of view to position the target.

2.4.4 Exposure Scheduling Constraints

TRANS gives SPSS information about scheduling constraints on alignments, obsets and scheduling units. TRANS determines whether they can be interrupted by the earth occulting the target, and what the maximum interruption duration is. TRANS determines, based on constraints from the proposal, whether internal alignments (those which do not point to an external target) can be executed when the telescope is changing its pointing. TRANS determines whether SPSS is free to interleave an obset's alignments those of another. TRANS chooses how large an area to

exclude for each alignment when scheduled near the South Atlantic Anomaly (SAA), the region in the southern hemisphere with too high a radiation level to accommodate certain instruments.

2.4.5 Commanding Parameters

TRANS computes many of the parameters necessary to command the telescope. The positions of the wheels which place filters, polarizer, gratings and apertures of the instrument field of view are computed by TRANS, determining if wheel motion is required. TRANS tracks of hundreds of commanding values, some specifically requested by the proposer and others it computes. TRANS also keeps track of the states of the instruments at the beginning and the end of each alignment and at what rate data is produced.

2.4.6 Computing Timing Relationships Between Objects

TRANS computes timing constraints which exist between alignments, obsets and scheduling units. Such timing relationships arise when an exposure in one higher level object (alignment, obset or scheduling unit) constrains an exposure in another. TRANS computes the minimum and maximum separation of alignments and obsets based on these constraints. When one scheduling unit constrains another, these minimum and maximum separations are noted in the form of a link between the two.

These timing constraints are computed using the same technology as ordering of exposures.

2.4.7 Other Values Computed By TRANS

TRANS computes a range of values required by PODPS to determine how to process data once observations are complete. These includes filter, grating, polarizer and aperture usage, as well as camera and spectrograph modes. Trans also performs computations on the target attributes provided by the proposer to make them suitable for use by SOGS.

2.5 Reporting the Results of Trans

Figure 2 shows the stages TRANS goes through to build its internal structure.

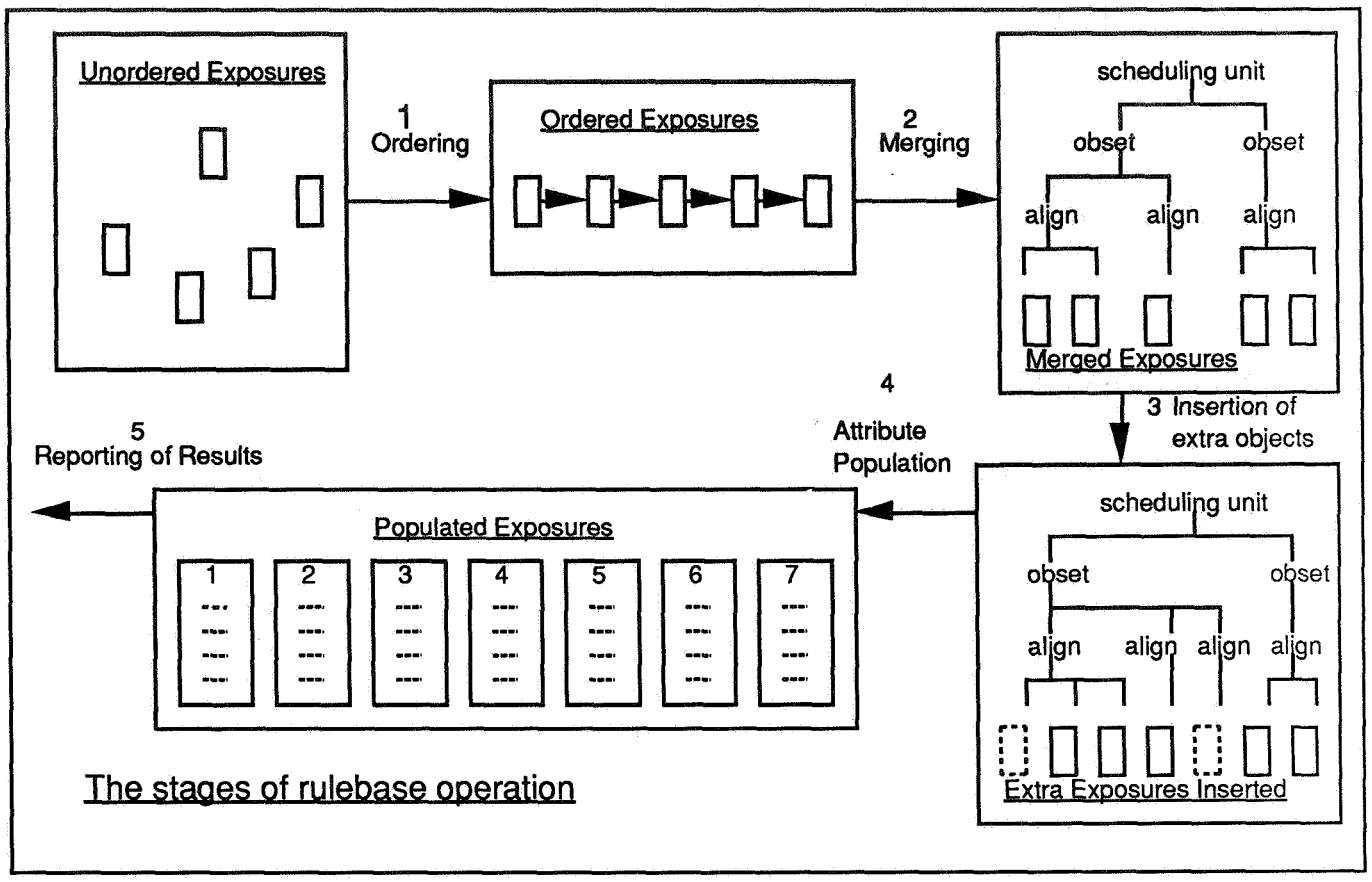

Figure 2

After this structure is build, TRANS outputs its conclusions. Though this is chiefly via text file, currently SPIKE is capable of operating in a mode where it can read results directly out of memory.

The largest output product of TRANS is the *assignment file*. Assignment files contain instructions in a database language (IQL or SQL) for populating the Proposal Management Database (PMDB), the chief source of information for SOGS. The assignment file can be edited to override TRANS decisions.

TRANS produces a record of its merging and ordering decisions in the *levels file*. A levels file can be edited and reloaded into TRANS to change the ordering or merging for a second run. In this mode, called *manual merge*, TRANS processes exactly as if it had automatically ordered and merged as specified in the edited levels file.

TRANS generates a *guide star request file* for each obset. Guide Star Requests can be fed into GSSS to determine whether guide stars are available for an obset. The output of GSSS is then used by SPIKE to determine schedulability of observations.

TRANS generates a *proposal auxiliary file* (PAF) for use by SPIKE. When an exposure has been identified by a proposer as conditional, it is recorded in the PAF. This file can be edited when it is determined whether to execute the observation.

TRANS generates a diagnostic file containing a list of problems. TRANS checks for cases it cannot handle and warns about observations that would be difficult or impossible to schedule (such as an alignment which could not be split, but is still too long for an orbit), observations that will present a problem for commanding and exposures that violate constraints of HST.

A series of reports can be generated to allow users to determine what decisions were made. There are several reports that document complicated decisions (such as merging) [Gerb89a].

The outputs of TRANS and their uses are illustrated in figure 3.

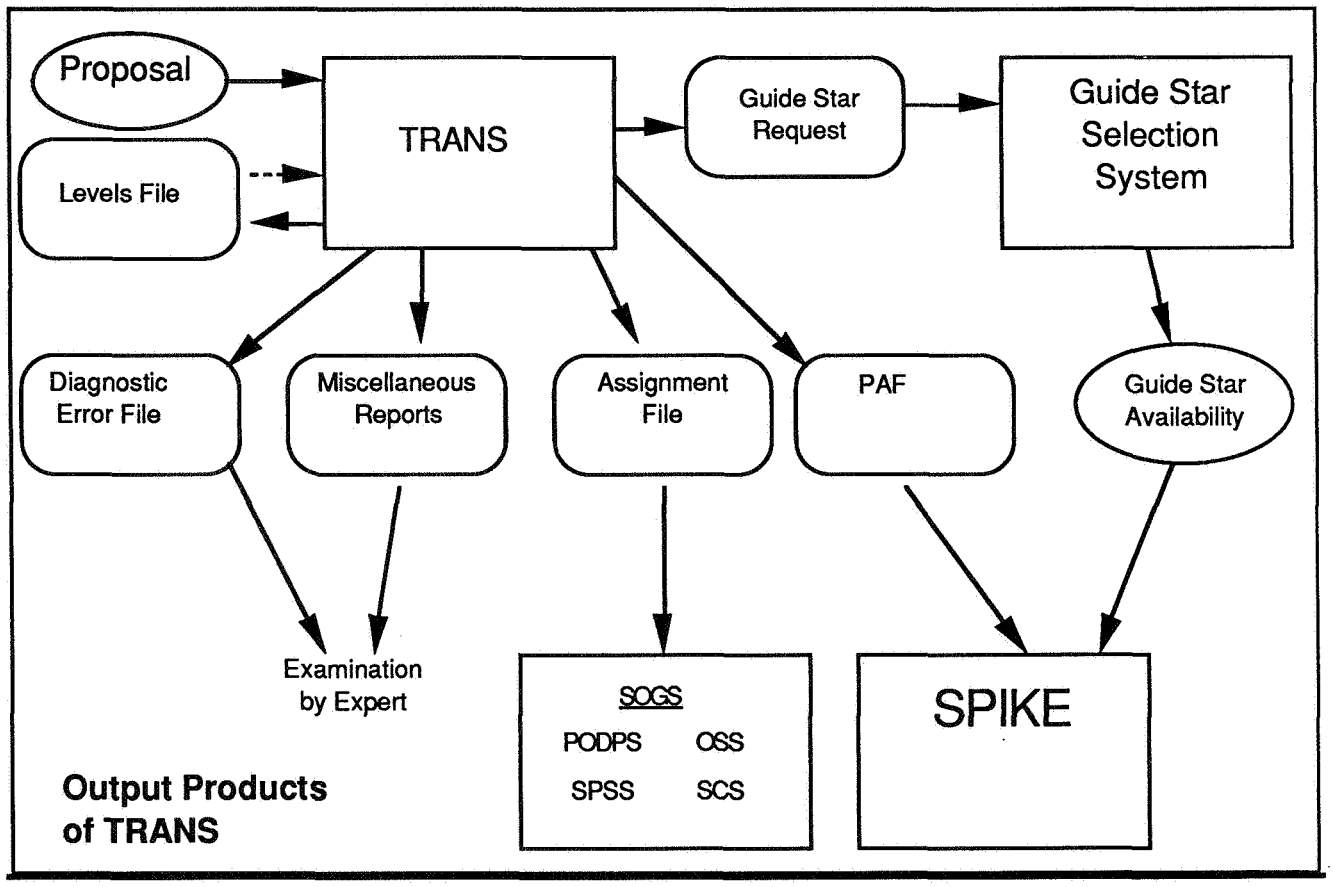

Figure 3

3. How TRANS is Implemented

Trans was originally built in 1985 using the OPS5 expert system definition language [Lindenmayer87a, Rosenthal86a, Rosenthal86b]. Although it originally appeared that TRANS solved a simple enough problem to be amenable to the use of OPS5, its subsequent complication

and the difficulties inherent in the OPS5 language have forced the conversion of TRANS into Common LISP.

3.1. The Implementation of TRANS in OPS5

OPS5 is a declarative language that allows the division of a task into *rules* [DEC89a]. Each rule consists of a left hand side containing conditions under which it should be executed, and a right hand side containing actions to be performed. Permissible actions involve the insertion or deletions of working memory elements (WMEs), the units of data used by OPS5, and the calling of outside functions. OPS5 uses a "bottom-up" order of execution, where a conflict set is accumulated of all rules whose left hand sides correspond with the current state of working memory. A disambiguation algorithm is used to determine which rule in the conflict set to execute. A new conflict set is then computed and the procedure repeats.

3.1.1. Problems with Declarative Properties of OPS5

The non-procedural nature of OPS5 made execution difficult to trace. When an undesired result (such as an output product not being correct) was encountered, it was very difficult to tell which rule caused the problem. Procedural languages allow for the placement of debugging output throughout the code to pinpoint the location of a problem. This approach does not work in OPS5, since it is impossible to look at a subset of the source code and determine which rule will execute next. It is necessary to understand the entire rulebase to predict the order of execution of rules. When the OPS5 version of TRANS had grown to over one hundred thousand lines in almost 1000 rules, this became a hopeless undertaking. To debug TRANS, developers were often forced to monitor the execution of rules using the trace facility provided by the environment, stopping execution periodically to determine if the anomaly had occurred. As a side effect, this characteristic of OPS5 forced developers to rely on language specific debugging tools far more than in a conventional language.

3.1.2. Deficiencies in the Rule Syntax

The syntax for defining the left hand side of OPS5 rules does not allow full context-free nested boolean syntax. The only condition allowed on the left hand side of an OPS5 rule is a conjunction of assertions of the existence or absence of a WME fitting certain specifications. For example the following condition could be expressed in a single OPS5 rule:

```
Execute if there exists a WME satisfying A and a WME satisfying
B and a WME satisfying C.
```

However, the following condition could not be expressed in a single OPS5 rule:

```
Execute if there exists a WME satisfying A and (a WME
satisfying B or a WME satisfying D) and a WME satisfying C.
```

This condition would have to be implemented by creating two rules with identical right hand sides and the following two left hand sides:

```
1. Execute if there exists a WME satisfying A and a WME
satisfying B and a WME satisfying C.

2. Execute if there exists a WME satisfying A and a WME
satisfying D and a WME satisfying C.
```

If a disjunction is desired in more than one element of the conjunction, there is a multiplicative proliferation in the number of rules needed. For example to express a conjunction of five disjunctions, each of which matched a WME of one of four different specifications, would require 1024 rules!

This scenario was played out over and over during TRANS development. Three major problems resulted from this. First, a requirement that was describable in English in a one line sentence, could translate into a dozen or more rules. Second, this problem often required TRANS developers to devote much time to figuring out the most efficient way to implement simple requirements. Often, no concise way could be found. Occasionally, shortcuts did allow fewer rules, but always at a cost of many hours of planning. Third, the abundance of nearly identical rules provided a constant temptation for developers to use cut-and-paste editing utilities to generate new rules. A minor typographical or logical error could be duplicated scores of times, creating future maintenance headaches.

Because of these problems, it became increasingly difficult to scope work estimates for TRANS projects. Some enhancements took ten times longer than originally expected because a rule explosion was encountered.

3.1.3. Lack of a Functional Capability

OPS5 lacks the ability to define procedures and functions. There is not even a macro substitution facility. The normal software engineering activity of providing a library of convenient procedures became impossible when operating in OPS5. One workaround involves the creation of "utility" rules, that populate special WME fields whenever such an operation is desired. This has the unpleasant side effect of greatly increasing the size of WME's and consequently the space required to run TRANS. This workaround does not give the capability within the execution of an OPS5 rule to execute a second rule, returning execution to within the first rule when the execution of the second is complete. OPS5 utterly lacks subroutine definition found in almost all modern languages.

The second workaround involved calling out to a functional language (in our case the C language) to execute function calls. The available interface between OPS5 and C was obscure and poorly documented and differed depending on whether the function was being called within the left hand side or right hand side of a rule and on what kind of data it processed. Calling C subroutines became such a frustrating and error-filled task, so that this workaround was chosen only as a last resort.

The main deficiency in OPS5 utilities was the lack of a string manipulation capability, complicating any requirement which required the examination of text.

3.2. The Implementation of TRANS in LISP

Starting in the fall of 1988, TRANS was reimplemented in Common LISP [Steele90a]. Using the LISP macro facility, a small but powerful extension to LISP was written called the transformation command language (XCL). XCL supports a procedural rule syntax and allows abstraction for underlying data structures. A shorthand for using these data structures was developed and a facility for generating simple reports and diagnostic errors was added.

3.2.1. Underlying Data Structures

The data structures in XCL allow for different object types. Each object type has a series of properties and keys defined for it [Johnston88a].

Objects were intended to model tuples in a relational database with properties corresponding to the fields of the relation. Definition of properties do not take up any memory space in the data structure unless they are populated with a non-null value. To this end, association lists were chosen to implement the underlying structures, instead of the existing LISP defstruct capability or an object-oriented system such as CLOS.

Keys consist of sequences of properties. For each key, XCL maintains a hash table, matching lists of values (each of which corresponds to a property in the key) to instances of the objects which contain them. In this way it is quickly possible to locate an instance of an object if the values of properties comprising a key are known.

XCL allows a shorthand to specify fields of data structures. It uses the LISP keyword package, choosing the period (.) to separate object instances from their properties. For example `:al.version_num` would correspond to the version_num property of the object denoted by :al. Indirect property reference is also permitted. `:al.exposure.si_used` refers to the si_used property of the object designated by the exposure property of the object :al.

XCL provides forms that allow iteration over a series of exposures. For example:

```
(FOR-EACH :qexposure :ex :do
  (assign :ex.pos_input_fg "N"))
```

sets the pos_input_fg property to "N" for every object of type :qexposure. This capability also allows iterating over all objects that match a certain key or are elements in a certain list.

XCL provides a type checking capability to make sure properties of objects are not referenced that do not exist.

3.2.2. Rule Syntax

XCL provides a form for the definition of rules [Johnston89a]. A rule definition is very similar to the defun form in LISP, except that no arguments are necessary. The following rule executes the fragment of XCL code used in the previous section:

```
(define-rule Set-position-input-flag
  "Sets pos_input_fg to N for each qexposure"
  (FOR-EACH :qexposure :ex :do
    (assign :ex.pos_input_fg "N")))
```

This rule is called Set-position-input-flag.

The body of a rule can contain any syntax legal in Common Lisp, as well as the syntactic extension provided by XCL. A facility also exists for defining functions with XCL object type arguments. The following function is an example.

```
(define-function Actual-Alignment-Time ((:al :qalignment))
  "Adds the time_require and acq_time fields of an alignment"
```

```
(+ :al.time_require :al.acq_time))
```

This function would take an XCL object type of :qalignment as an argument. Run time type checking is performed to insure that the argument is of the proper type.

Rules are executed in the order that they appear in their source file. Source files are grouped into *phases*, which are a series of rules, usually executed together. When a phase is defined, the order of execution of its source files is provided. Phases can also be defined to contain subphases, so that the execution of a phase can involve the execution of one or more lower level phases. Typically, TRANS is executed by invoking a phase containing subphases with all the rules to be executed.

Figure 4 illustrates the relations ship between phases, subphases, source modules and rules.

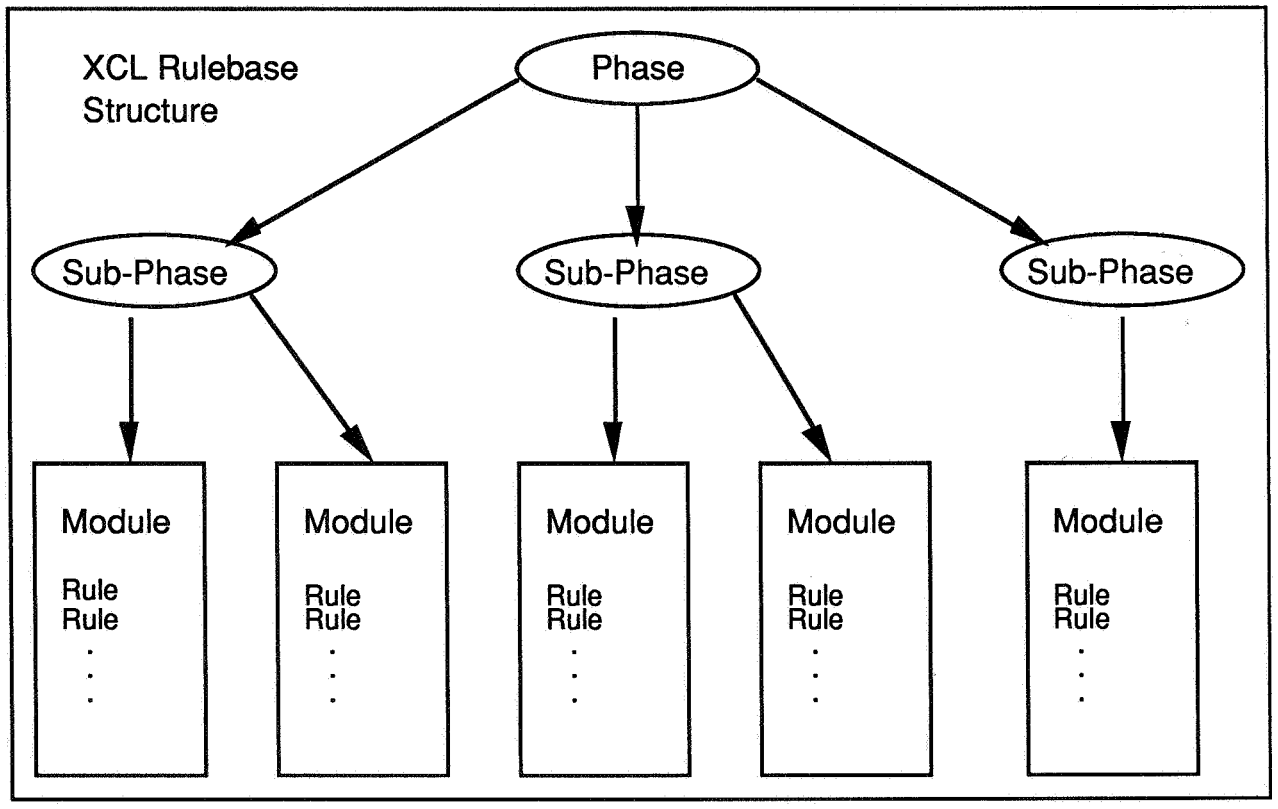

Figure 4

3.2.3. Other Features of XCL

XCL allows a generator for custom-defined reports. Currently it allows an XCL developer to report over a given object type, outputing one record per instance. Records consist of output fields which corresponds to properties of the object, or computations on it, and may span more than one line. Report fields can be formatted using any of the capabilities available in the Common LISP format statement. Output can be ordered according to keys defined on the output object. Report headings can be specified and one or more title lines included. A planned future

enhancement will allow the specification of selection criteria which will exclude any objects from the report.

XCL also provides a table lookup facility where tables are included as part of the XCL source and can be referenced by one or more keys. There is also a flexible diagnostic error generation facility which maintains statistics on the frequency of various problem types.

4. Conclusion

Since the conversion of TRANS from OPS5 to XCL, the turnaround time for enhancements has been reduced greatly. As a result, numerous capabilities have been added. The generation of Guide Star Requests, the expanded ability to output diagnostics and the more complex transformations required to create the TRANS output structures are examples of capabilities that the XCL version of trans performs as a matter of course but would have caused major problems in the OPS5 version. Requirements which used to generate numerous rules with over hundreds of lines can now be implemented far more comprehensibly in one rule with only a few lines.

Three major reasons exist for the improved turnaround time. First, the rulebase can be maintained in a modular hierarchical way, making it much easier to understand the order of execution of rules and phases. Second, its procedural nature facilitates debugging of the rulebase - an anomaly can be systematically pinpointed without lengthy tracing of execution. Third, the nature of XCL and its LISP underpinnings have allowed TRANS developers to create a rich functional base, providing all the software engineering benefits of code reusability.

We are now finding that non-developers are occasionally able to peruse the TRANS rulebase and determine its functionality, reducing the need for developers to perform a user support function. This activity was strongly encouraged while the OPS5 version was being maintained, but did not occur in practice until TRANS was reimplemented using XCL.

Three major lessons lessons were learned from this project. First, that as an expert system definition language, OPS5 has deficiencies which intensify as the problems it is applied to become increasingly complex. Second, that it is worth considering a procedural knowledge representation - that such a representation is easy to build, comprehend and modify. Third, that Common LISP provides an ideal platform for such a procedural expert system shell because of its versatility, extensibility and wealth of predefined utilities.

The successful re-implementation of TRANS has made it one of the most flexible elements of the HST ground software systems. Because it provides the "bridge" between observing programs and flight operations, it provides the key point of leverage when changes in observing strategies arise. Work is underway to define how TRANS can further optimize the efficiency of HST, which should make available hundreds of hours per year of valuable observing time that would otherwise be lost, Finally, because of the extensive observation checking performed by TRANS, plans are being mde to provide a distributable version of TRANS that could be used by astronomers at their home institutions to help prepare observing programs

Acknowledgements

Implementation and maintenanceof the OPS5 version of TRANS involved Don Rosenthal, Kelly Lindenmayer, Jim Sims, Bob Jackson and Andy Gerb. Implementation of the LISP version of TRANS involved Mark Johnston, Bob Jackson, Kelly Lindenmayer, Jim Sims, Andy Gerb, Mike Rose and Ron Henry. Requirements definition for TRANS was aided by Sid Parsons, Doug McElroy, Eliot Malumuth, Mike McCollough, Jim Roberts, Jim Mainard and John Baum.

References

DEC, 1989, "VAX OPS5 User's Guide", Digital Equipment Corporation [DEC89a].

Gerb, A., 1989, "The Transformation User's Manual", SPIKE Technical Report Number 1989-13, Advance Planning Systems Branch, Space Telescope Science Institute, [Gerb89a].

Gerb, A., 1990, "Transformation Design Manual", SPIKE Technical Report Number 1990-2, Advance Planning Systems Branch, Space Telescope Science Institute, [Gerb90a].

Jackson, R., Johnston, M., Miller, G., Lindenmayer, K., Monger, P., Vick, S., Lerner, R. and Richon, J. 1988, "The Proposal Entry Processor: Telescience Applications for Hubble Space Telescope Science Operations", Proceedings of the 1988 Goddard Conference on Space Applications of Artificial Intelligence [Jackson88a].

Johnston, M., Lindenmayer, K., Sims, J. and Gerb, A., 1988, "Transformation System Programmer Manual", SPIKE Technical Report Number 1988-7, Advance Planning Systems Branch, Space Telescope Science Institute, [Johnston88a].

Johnston, M., Lindenmayer, K., Sims, J. and Gerb, A., 1989, "Transformation Script Programmer Manual", SPIKE Technical Report Number 1989-8, Advance Planning Systems Branch, Space Telescope Science Institute, [Johnston89a].

Lindenmayer, K., Vick, S. and Rosenthal, D. 1987, "Maintaining and Expert System for Hubble Space Telescope Ground Support" in Proceedings of the 1987 Goddard Conference on Space Applications of Artificial Intelligence and Robotics [Lindemayer87a].

Miller, G., Rosenthal, D., Cohen, W. and Johnston, M. 1987, "Expert Systems Tools for Hubble Space Telescope Observation Scheduling" in Proceedings of the 1987 Goddard Conference on Space Applications of Artificial Intelligence and Robotics, reprinted in Telematics and Informatics, 4, 301-311 [Miller87a].

Rosenthal, D., Monger, P., Miller, G., and Johnston, M., 1986, "An Expert System for Ground Support of the Hubble Space Telescope", Proceedings of the 1986 Goddard Conference on Space Applications of Artificial Intelligence and Robotics [Rosenthal86a].

Rosenthal, D., "Transformation of Scientific Objectives into Spacecraft Activities", 1986, Expert Systems in Government Symposium [Rosenthal86b].

Space Telescope Science Institute, 1989, "Hubble Space Telescope, Phase II Porposal Instructions", Space Telescope Science Instute [Space89a].

Steele, G., 1990, "Common LISP, The Language, Second Edition", Digital Press [Steele91a].