NASA Technical Memorandum 100613

STRUTEX
A PROTOTYPE KNOWLEDGE-BASED SYSTEM FOR INITIALLY CONFIGURING A STRUCTURE TO SUPPORT POINT LOADS IN TWO DIMENSIONS

James L. Rogers, Stefan Feyock,
and Jaroslaw Sobieszczanski-Sobieski

(NASA-TM-100613) STRUTEX: A PROTOTYPE KNOWLEDGE-BASED SYSTEM FOR INITIALLY CONFIGURING A STRUCTURE TO SUPPORT POINT LOADS IN TWO DIMENSIONS (NASA) 24 p
CSCL 09B G3/61

N88-23448

Unclas
0142702

April 1988

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

STRUTEX
A PROTOTYPE KNOWLEDGE-BASED SYSTEM FOR INITIALLY CONFIGURING A STRUCTURE TO SUPPORT POINT LOADS IN TWO DIMENSIONS

J. Rogers[1], S. Feyock [2], J. Sobieszczanski-Sobieski[1]

[1] NASA Langley Research Center, Hampton, Virginia, USA

[2] VAIR, Inc., Williamsburg, Virginia, USA

ABSTRACT The purpose of this research effort is to investigate the benefits that might be derived from applying artificial intelligence tools in the area of conceptual design. Therefore, the emphasis in this paper is on the artificial intelligence aspects of conceptual design rather than structural and optimization aspects. A prototype knowledge-based system, called STRUTEX, has been developed to initially configure a structure to support point loads in two dimensions. This system combines numerical and symbolic processing by the computer with interactive problem solving aided by the vision of the user by integrating a knowledge base interface and inference engine, a data base interface, and graphics while keeping the knowledge base and data base files separate. The system writes a file which can be input into a structural synthesis system, which combines structural analysis and optimization.

One objective was to investigate methods for passing data between a data base and a knowledge base. This was accomplished by separately integrating two types of inference engines, one forward chaining based on production rules, and one backward chaining based on PROLOG, into the system and determining their effects on the flow of data between the knowledge base and the data base. No significant problems were encountered in integrating either of the inference engines. Nor did one inference engine run significantly faster than the other for this small knowledge base. It was concluded that these two systems supplement rather than compete with one another, and further research is warranted to investigate the simultaneous integration of both inference engines into the system and determine the effects on conceptual design.

A second objective was to examine when it is preferable for a computer to supply the data and when it is preferable for the data to be supplied by human vision. It was also concluded during the development of this system, that there are times to rely on the computer and there are times to rely on the vision of the user. For small problems such as the ones used for testing, there were several instances where the user's vision was more preferable than relying on the computer, such as determining the location of the support surface relative to the loads. However, for larger, more complex problems, it might be preferable to add symbolic rules to the knowledge base, numerical algorithms to the main program, and rely on the computer. More research is also needed in this area.

INTRODUCTION

Engineers and management are always concerned about reducing the costs and time involved in completing a design project. Therefore, many hours of research have been devoted to speed and sensitivity improvements in the area of structural analysis. Additional research effort has been applied to the improvement of optimization algorithms. From a numerical standpoint, these areas are nearing a point of diminishing returns when using conventional computer hardware. However, one area which shows a potential for reducing design cost and time, but has had little research, is the determining and refining of an initial configuration before beginning the analysis and optimization process (figure 1). One reason is because this is a problem that is not easily solved numerically, but one that seems to require using heuristics from experienced designers.

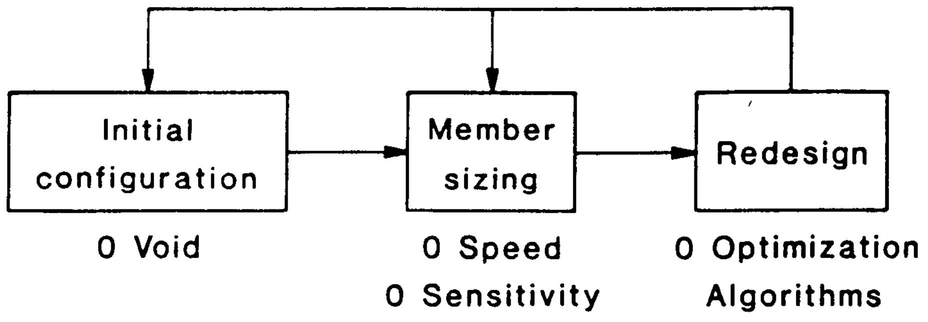

Figure 1. Research efforts in design.

Several years ago, engineers began applying artificial intelligence (AI) tools and techniques to problems in different engineering disciplines. There are excellent overviews of these applications by Sriram [16,17]. Next, these tools were applied to optimization problems by Baenzinger [1], Chieng [3], and Rogers [14]. The use of AI tools was expanded to applications in the decision making processes of design by Gero [8] and Mistree [10]. Only recently have engineers begun making use of the AI tools in the area of conceptual design by MacCallum [9] and Shah [15].

To continue filling this void in the design process, a prototype knowledge-based system, called STRUTEX, has been developed to initially configure a structure to support point loads in two dimensions. This prototype was developed for testing the application of AI tools to conceptual design as opposed to being a testbed for new methods for improving structural analysis and optimization. This system combines numerical and symbolic processing by the computer with interactive problem solving aided by the vision of the user. One of the objectives of this project was to investigate methods for passing data between a data base and a knowledge base. Therefore, both the data and the knowledge (rules) are kept in different files separated from the main program. The other objective was to examine when it is more appropriate to rely on the computer for data and when it is more appropriate to rely on the vision of the user.

This paper describes how the system is constructed to interact with the user. Of special interest is the information flow between the knowledge base and the data base under control of the algorithmic main program. This paper also examines the trade-off in computing data within the program versus entering data interactively after making a visual determination. Examples of computed and refined structures are presented during the explanation of the system. Plans for enhancements to the system and conclusions are discussed.

OPERATION OF THE SYSTEM

STRUTEX emulates an engineering student taking a blank sheet of paper to a teacher to discuss an idea for building a structural model to support one or more point loads in two dimensions. As the teacher asks questions about the loading conditions and the support surface, the student responds with answers or by sketching ideas on the piece of paper. Based on what is seen and heard the teacher can help the student determine a reasonable initial structure for supporting the given loads. In STRUTEX, a knowledge base replaces the teacher, a graphics window on the computer replaces the piece of paper, and a dialog area in the graphics window replaces the verbal question and answer. The user interactively interfaces with the system through two methods, typed dialog and mouse-oriented graphics. The user graphically inputs loading and support surface data using the mouse in response to questions from the system. The user also types in responses to system questions about the load points, support surface, and support structure. The data is stored in a relational data base.

Once all questions are answered, the appropriate data is transferred from the data base to the knowledge base and the system determines the type of structure most suitable for satisfying the input conditions. If the structure is determined to be a beam(s) or a string(s), then the structure is drawn on the graphics window and the session is completed. If there is only one load and the structure is determined to be a truss, then other rules are invoked to determine whether or not bracing is needed, and, if so, the type and amount of bracing. This structure is then drawn on the graphics window. If there is more than one load point and the structure is determined to be a truss, then the user is guided by recommendations in a step-by-step approach to building the truss. The truss built by the user is then tested against rules in the knowledge base and recommendations are given for the user to improve the model. This is done iteratively until all rules are satisfied and no recommendations for improvements are made. An input file for a structural analysis program is written for a truss so that the model can be analyzed and optimized by a previously developed system for structural synthesis.

COMPONENTS OF THE SYSTEM

The main driver program for STRUTEX is written entirely in FORTRAN. Other components were added by linking existing software - DI-3000 [5] for the graphics, RIM (Relational Information Management, Erickson [6]) for the relational data base management, and CLIPS (C Language Production System, Riley [12] for the inference engine - to the main driver program. The data for RIM and the knowledge base (rules) for CLIPS are maintained in different files separated from STRUTEX. EAL (Engineering Analysis Language, Whetstone [21]) for the structural analysis, and CONMIN (Constraint Minimization, Vanderplaats [20]) for the optimization are coupled in PROSSS (Programming System for Structural Synthesis, Rogers [13]) to perform the analysis and optimization (figure 2).

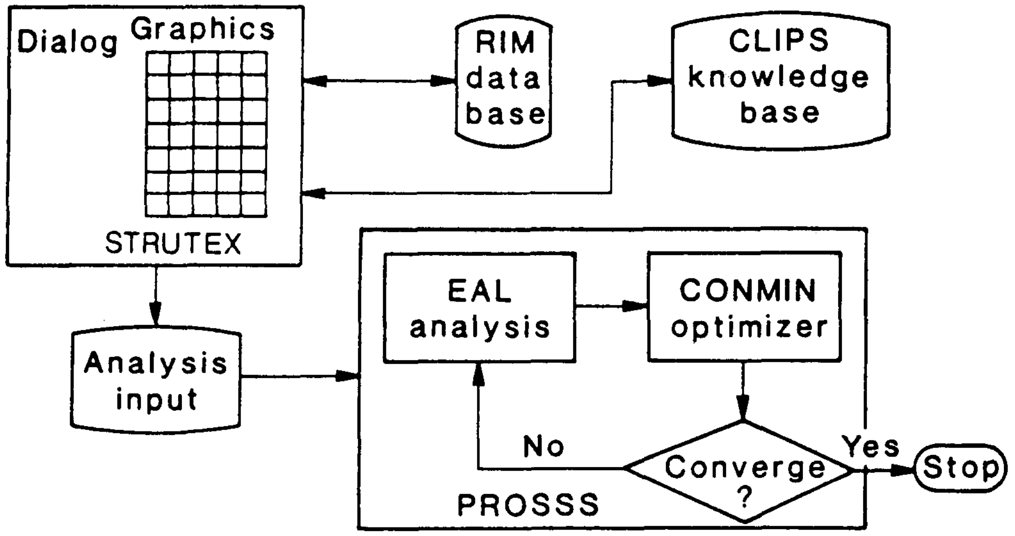

Figure 2. Diagram of STRUTEX with PROSSS.

Graphics

The DI-3000 graphics package is a device independent system and is the primary graphics system used at NASA Langley Research Center. However, in STRUTEX all calls to DI-3000 are made from a single routine making it simple for a user to replace DI-3000 with another graphics package. Graphics calls include moving the cursor, drawing lines and circles, text, and receiving data from the mouse.

The graphics window is divided into two parts, a graphics part with a grid to aid the user in drawing the structure and a dialog area for the user to receive questions and type responses. The mouse is used to place the crosshairs on a particular point to input a piece of data.

Data base

RIM is a relational data management system used for storing data about the structure. Like the graphics part of the system, all calls to RIM are made from a single routine to make it simple for the user to replace RIM with another data management system. Data base calls include opening and closing the data base, finding a relation, loading data, retrieving data, and deleting data.

STRUTEX uses three relations to store the data. The load relation contains the load number, type of load (ex. gravity load), X-Y coordinates of the load point, magnitude of vertical and horizontal loads, and the distance from the load to the support surface. The surface relation contains the location of the support surface with respect to the load, the X-Y coordinates of the endpoints and midpoint of the support surface, and the area of the support surface if it is not a point. The support relation contains the member number, the type of support (ex. truss), the weight of the support, the X-Y coordinates of the endpoints of the member, and the length of the member.

Analysis and optimization

Once the conceptual design of the truss has been completed, an input file for the EAL structural analysis code is written. The user is questioned about the initial design variables which are the cross-sectional areas of the members. The user must also input the upper and lower bounds for these variables. The final input consists of the material properties. The properties and allowable stress for aluminum and steel are coded into the system. Users can choose one or the other, or users can input their own material properties and allowable stress. The program ends and the user can execute PROSSS. PROSSS, which uses a finite difference technique, will loop between EAL and CONMIN until the objective function, weight for this problem, is minimized or the system terminates because the maximum number of iterations has been reached without minimizing the objective function.

FLOW OF DATA BETWEEN THE KNOWLEDGE BASE AND THE DATA BASE

Data base systems typically have little knowledge, much data, and rely on fast secondary storage techniques. Knowledge- based systems, on the other hand, have much knowledge, little data, and work within main memory. If knowledge-based systems are to be integrated into the design process, new methods must be developed so that they can handle the large amounts of data typically created during a design project.

There are three approaches in current long-term research efforts which are trying to determine the best way to couple these two types of system into a single system with the best features of both, Nguyen [11], van Biema [19], Ceri [2]. One approach is to begin with an existing data base system and add knowledge base features to it. A second approach is to begin with an existing knowledge-based system and add data base features to it. The third approach, and probably the most promising, is to start from scratch and develop a completely new system combining the best features of both data and knowledge-based systems. The best short-term solution appears to be taking an existing knowledge-based system and an existing data base system and coupling them with an interface such as the one used by Feyock [7]. This is the approach taken with STRUTEX.

The rules for STRUTEX are very simple and could probably have just as easily been incorporated into the main driver program with IF-THEN statements. However, one of the objectives of this project was to investigate methods for passing data between a data base and a knowledge base. Therefore the knowledge base and the data base are maintained in different files separate from the main program. The interface is made through STRUTEX by linking a RIM interface library and a knowledge base interface library with STRUTEX. If data is needed from the knowledge base, a rule or rules must be executed. If the data is not available within STRUTEX, it is retrieved form the data base by calling RIM interface subroutines. That data is then asserted into the knowledge base again using interface subroutines. Data is returned from the knowledge base to STRUTEX which, in turn, stores the data in the RIM data base if it is necessary.

To better understand the interface between the knowledge base and the data base, two complétely different types of knowledge-based systems were integrated into STRUTEX. One system is forward chaining and based on production rules, while the other system is backward chaining and based on PROLOG.

A production rule knowledge base/inference engine

CLIPS is a knowledge-based system tool developed at NASA Johnson Space Center. It is written in C, performs forward chaining based on the Rete pattern matching algorithm, and has a FORTRAN interface. The knowledge base is composed of rules which are defined by the "defrule" construct. A rule states specific actions, the Right-hand side (RHS), that are to be taken if certain conditions, the Left-hand side (LHS) are met. An "= >" separates the LHS and the RHS. If and only if all of the conditions on the LHS are satisfied, then the actions on the RHS are performed sequentially. Each rule must contain at least one condition and one action, however there is no upper limit on either the number of conditions or actions.

Pieces of information represented by facts, the basic form of data in CLIPS, are contained in a facts-list. A fact is composed of several fields with each field being separated by a space. A field can contain a number, a word, or a string. Facts are asserted into the facts-list before execution by the "deffacts" construct or by an assert command in the calling program, or during execution as the action caused by executing a rule. A rule executes based on the existence or non-existence of facts in the facts-list. For example, the rule for selecting a string as the type of support is:

```
(defrule string
(SURFLC ABOVE)
(PLOADT GL) =>
(assert (SUPPORT STRING))
(KBANS1 SUPPORT STRING 0.0))
```

This is read: If the location of the support surface is above the load and the load is a gravity type of load then the support type is a string. This rule will execute when the two facts (SURFLC ABOVE) and (PLOADT GL) are asserted from STRUTEX and placed into the facts list. The actions, based upon a match on the two facts (conditions), are to return to STRUTEX via the KBANS1 parameter the fact that the support is a string, and to assert the fact that the support is a string into the facts-list. The KBANS1 parameter, discussed below in more detail, is the name of the subroutine in STRUTEX. In this example, only the parameters SUPPORT and STRING are needed. The 0.0 is a dummy parameter.

Currently there are only thirteen rules in the knowledge base. There are three rules for determining the type of support, beam, truss, or string. The rules for choosing the beam or truss are more complex than that of the string and use an explicit "or" coupled with three or four explicit "and"s. Another rule in the knowledge base explains the choice of support type when executed. The remainder of the rules determine whether or not bracing is required and the type of bracing that is required in a truss by comparing the length of members and checking certain angles the members make with each other. A simple structural analysis is used to create facts for the rule to determine the type of bracing based on the length of the members in ratio to the loads. The type of bracing is chosen between a "Z" or "V" type. Other rules determine if there any angles formed which are greater not within a given range. If so, a recommendation is made to correct the problem. The action parts of the bracing rules are more complex than those of the rules for choosing the type of support. Some of the bracing actions are based on mathematical computations within the rule, while others have choices of actions within a single rule with the choice being determined by the facts.

The inference engine in CLIPS applies the knowledge (rules) to the data (facts). Pattern matching occurs on the LHS for single- and multiple-fields, single- and multiple-field wildcards, and single- and multiple-field variables. The basic execution cycle begins by examining the knowledge base to see if the conditions of any rules have been met. All rules with currently met conditions are pushed onto the "agenda" which is essentially a pushdown stack. Once the agenda is complete the top rule is selected and the RHS is executed. As a result of these actions, new rules can be placed on the agenda and rules on the agenda may be removed. This cycle is repeated until all rules that can execute have done so.

STRUTEX has three subroutines which interface with CLIPS by calling subroutines in the CLIPS FORTRAN interface library. Subroutine KBXEC initializes CLIPS, loads the rule base, and is called by other subroutines to assert facts into the knowledge base and execute the inference engine. Once all of the rules on the CLIPS agenda have been executed, control is returned to this subroutine which allows STRUTEX to continue processing. The other two routines, KBANS1 and KBANS2, receive data from CLIPS after the appropriate rule (or rules) has been executed. These two subroutines are called as an action in CLIPS. KBANS1 has three parameters, two alphanumeric and one numeric, while KBANS2 has three numeric parameters. The data returned from CLIPS are stored in these parameters for later use in other subroutines in STRUTEX.

A PROLOG-based knowledge base/inference engine

To better understand the flow of data between the knowledge base and the data base, CLIPS was removed from the system and replaced by another type of knowledge-based system. The replacement knowledge-based system is based on the PROLOG programming language Clocksin [4], the most important implementation of logic programming. While CLIPS operates in the forward chaining mode, the PROLOG inference engine is based on Horn clause resolution theorem proving. Since PROLOG operates as a backward chaining system, it requires a programming style that is largely declarative rather than procedural.

The PROLOG implementation is the University of York interpreter by Spivey [18], which is written in PASCAL. An interesting and powerful feature of this and other PROLOG implementations is the fact that the main interpreter loop is written in PROLOG rather than being embodied in the interpreter's PASCAL code. The actual processing of user input is performed by PROLOG rules which define the PROLOG procedure $top and are read when the interpreter is initialized. This PROLOG code, which effectively defines the PROLOG runtime system, is thus accessible to, and modifiable by, the user.

To allow PROLOG to be called in embedded mode, it was sufficient to make the following modifications:

1. make the interpreter code that calls $top available to outside programs as an external procedure
2. add two new built-in predicates,import and export, to the interpreter. import(X) retrieves data which the non-AI calling program has placed in a common area and binds it to X; export(X) places the data bound to X in the same common area for the calling program to find
3. change the PROLOG rules defining $top as follows:
   $top :- import(X), process(X).
   The process(X) procedure is defined by the user to perform any desired AI processing, and return to the caller after completion. In this case,

process(X) triggers a set of rules concerned with conceptual design.

As with CLIPS, the user inputs data such as the number of loads, the type of load, the load magnitude, and similar information. All of this data is stored in the RIM database. The knowledge base is then executed to determine the type of support that is required. The PROLOG knowledge base contains the same knowledge as is found in the CLIPS knowledge base, only in a different format. For example, here is the PROLOG rule similar to the above CLIPS rule for selecting a string as the support type:

```
string :- surflc(ABOVE), ploadt(GL).
/* a string support is appropriate if the support surface location is above
the load and the type of load is gravity*/
```

As is the case with the CLIPS-based system, features of the design are checked against the knowledge base and recommendations for improvements are made. The interaction continues until the user is satisfied. For more details on this knowledge base and inference engine, the reader is referred to Feyock [7].

Comparison of the two systems

The chief distinction between the PROLOG knowledge base and the CLIPS knowledge base is the underlying model. A forward chaining system is based on production rules which monitor database updates, and perform the actions stipulated in their RHS, generally further database updates, if their antecedents are satisfied. PROLOG, on the other hand, operates on the basis of logical proofs. The above rule, for example, indicates that to prove that a string support is indicated, it is necessary to prove the subgoals surflc(ABOVE) and ploadt(GL).

From the preceding discussion, it is evident that while CLIPS and PROLOG rules may have a superficial similarity, the operations of their respective inference engines are very different. It is therefore appropriate to note that these systems supplement rather than compete with one another. A production system, like CLIPS, is preferred when the programmer wants to retain a significant degree of procedural control of the computation, but requires flexibility and pattern-matching beyond those that FORTRAN can easily provide. Logic programming, on the other hand, allows for a declarative programming style, and furnishes the programmer with the great power of a backtracking theorem prover. Either or both of these systems can be integrated into STRUTEX and invoked as required. This is possible within STRUTEX because the database and the knowledge base are completely separate, and the knowledge bases have separate interfaces for passing data to and receiving data from STRUTEX.

USER INTERACTION WITH STRUTEX

When STRUTEX begins execution, the user first answers questions about the loads. The number of load points is the first input. The next input is the type of load which can be a gravity load, vertical load, sideways load, or a combination of gravity or vertical and sideways. This is followed by an iterative process through the load points where the user inputs the vertical and horizontal magnitudes of each load and then uses the mouse to locate the load point on the graphics window. Since no units are required by STRUTEX, the user must determine the correct units for the distances and loads.

The second stage of user input concerns the support surface. The system must know where the support surface is in relation to the load points - above, below,or to the side. This is an area where a computation could be done to determine the position, but it is much easier to let the user make this determination visually. The user then uses the mouse to place the midpoint of the support surface on the graph. The distance from the support surface to the first load is input without units. The distances from the remaining loads to the support surface are computed. The user inputs whether or not the support surface is a point. If it is not then the length of the support surface is input, again with no units.

The final piece of data that is needed before the system can determine the type of support is whether or not the support must be lightweight. Once this data is known, facts are asserted into the knowledge base and the inference engine executes the rules. The type of support is returned from the rules and stored into the data base. The choice and explanation of that choice are displayed on the dialog screen such as:

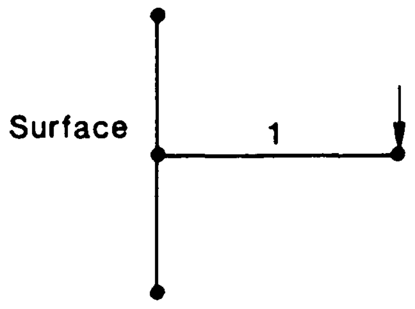

Figure 3. Beam supporting a single load.

A truss is the choice for a support.
****

Reasons:
The support surface location is to the side of the loads.
The support surface is not a point.
The support must be lightweight.

If the choice is a beam or a string the system draws the support (figure 3) and that ends the program. Currently, no input file for the structural analysis program is written for the beam or the string. If the choice is a truss and there is only one load point, a triangular structure is drawn (figure 4a). The system then determines whether or not bracing is needed by checking the ratio of the forces in the members against the length of the members (equation 1). The forces are computed from equation 2 representing static equilibrium of the loaded point.

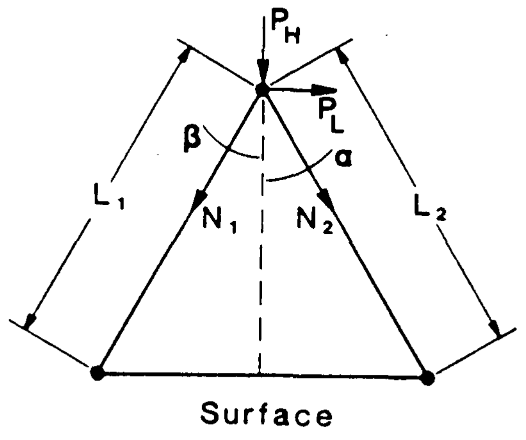

Figure 4a. Truss to support a single load checked by simple analysis

$$N1/L1 \geq \text{TOLERANCE} \qquad N2/L2 \geq \text{TOLERANCE} \qquad (1)$$

$$\begin{aligned} (N1)(\cos\beta) + (N2)(\cos\alpha) - P &= 0 \\ (N1)(\sin\beta) - (N2)(\sin\alpha) + P &= 0 \end{aligned} \qquad (2)$$

Facts are asserted into the knowledge base. The inference engine executes the rules and the choice is returned to the main program. If bracing is needed, the two side members are divided, and depending on the angle Delta, either a "Z" brace (figure 4b) or "V" brace (figure 4c) is chosen by the knowledge base. An input file is created for the analysis program and the program ends.

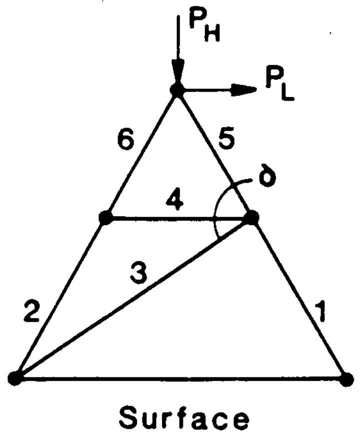

Figure 4b. "Z" bracing of truss to support a single load if $\delta > 40^0$.

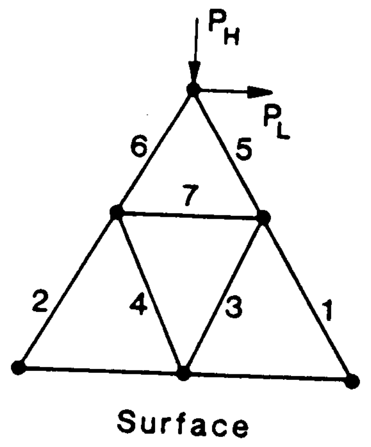

Figure 4c. "V" bracing of truss to support a single load if $\delta \leq 40^0$. (see fig. 4b)

If the choice is a truss and there is more than one load point, then the user must build the truss guided by recommendations from the system. The user begins with the load points and the support surface (figure 5a).

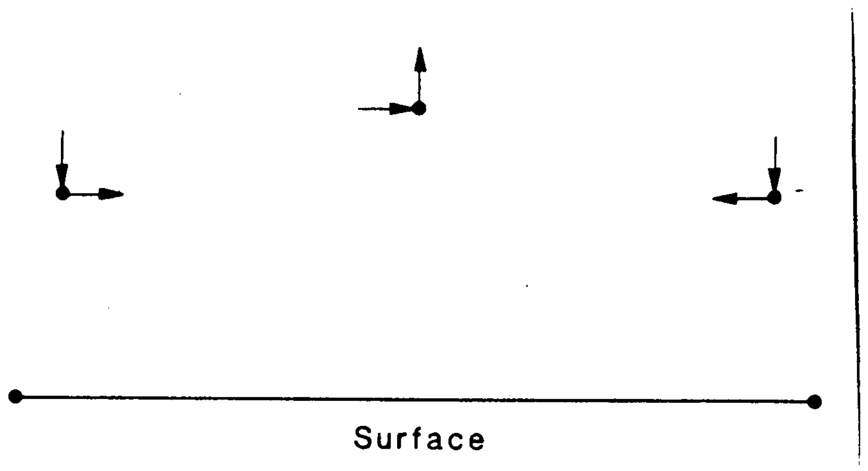

Figure 5a. Starting point to build a truss to support multiple load points.

A recommendation is made to connect all the load points forming triangles whenever possible, but not connect the load points to the support surface. Using vision rather than the computer, members are added by placing the mouse on the end points (figure 5b).

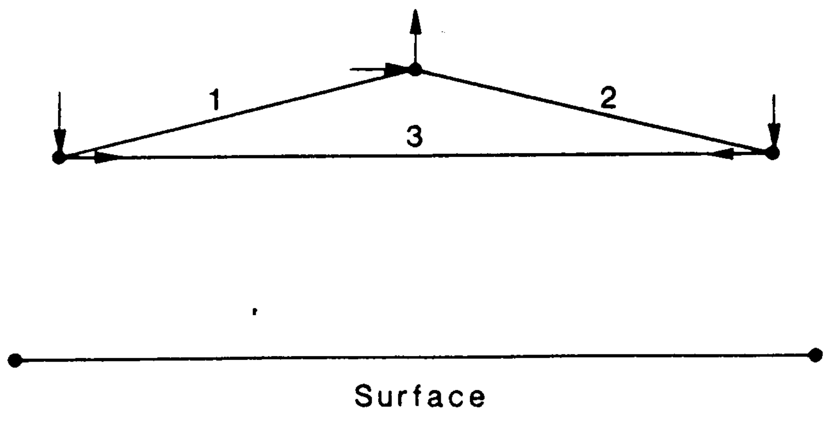

Figure 5b. Load points connected to form a triangle.

Once this has been completed, there is a second recommendation for the user to connect the load points to the support surface without having a new member intersect an existing member (figure 5c). (The reason for using two recommendations to build the truss is discussed below.) This step is also accomplished with the aid of the user's vision.

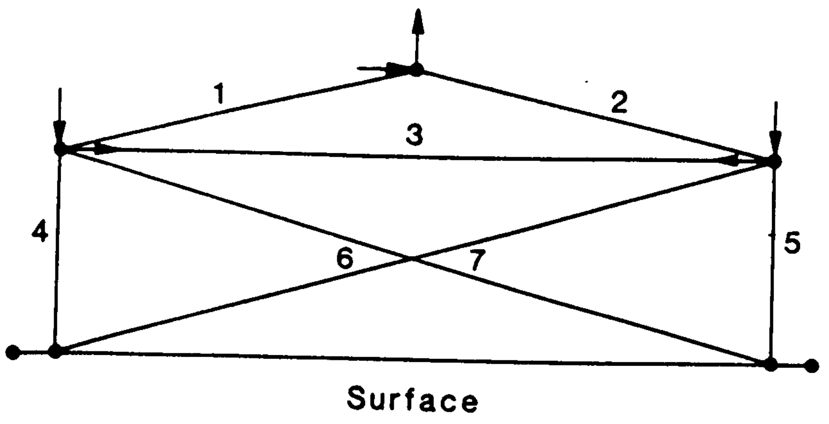

Figure 5c. Load points connected to support surface.

After this step is complete, the system determines all the triangles formed by the members and checks their angles. If the knowledge base finds that there are angles in the model outside a given range a recommendation is made to correct the problem. The limiting values for the angles are judgmental and can be changed based on the experience of the user. An example of such a recommendation based on the angles in figure 5c is:

******RECOMMENDATIONS******
The following triangles contain angles that are less than 15 degrees, therefore a modification may be required.

TRIANGLE	ANGLE	OPPOSITE MEMBER
1 2 3	13.7	1
1 2 3	12.4	2

```
            N1
            /\
           /  \
      N2/      \N3
```

If two external members form the angle then to expand the angle

(1) Remove the two members N1-N2 and N1-N3 that form the angle.

(2) Add two new members N1-N4 and N1-N5 to form a larger angle.

(3) Add a new member to connect N4 and N5.

(4) Add two members to connect N2-N4 and N3-N5.

```
        N1
        /\
       /  \
   N4/____\N5
     |      |
   N2|      |N3
```

If this recommendation can be implemented in more than one way, choose the way that will contract the structure, rather than expand it.

```
*******************************************
The following triangles contain angles that are greater than 120 degrees,
therefore a modification may be required, such as adding a new member to
divide the angle into two smaller angles.
    TRIANGLE      ANGLE      OPPOSITE MEMBER
    1 2 3         154.                3

*******************************************
```

The user then removes all members which contribute to the problem. This is another point where the user's vision can aid in determining which members to remove. The user then adds new members to satisfy the recommendation. If the user desires, the angles in the model can again be checked for problems. This is repeated until the user is satisfied (figure 5d). The input file for the structural analysis program is written and the program ends.

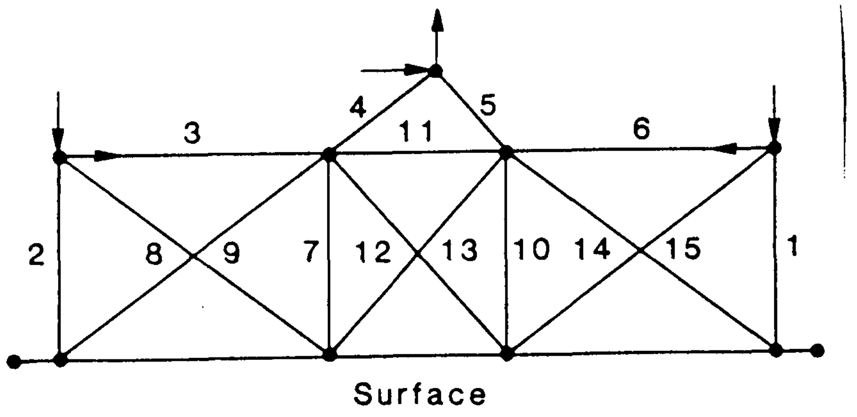

Figure 5d. Refined truss based on recommendations.

For the truss with multiple load points, there are two recommendations instead of one to allow the knowledge base to determine the bracing required between two members connecting a load point to the support surface. When two members connecting a load point to the support surface form a quadrilateral, the knowledge base is given the lengths of the two members. If the two lengths are the same, an "X" bracing is added. If the two lengths are different, a brace is added from the bottom of the longer member to the top of the shorter member (figure 6).

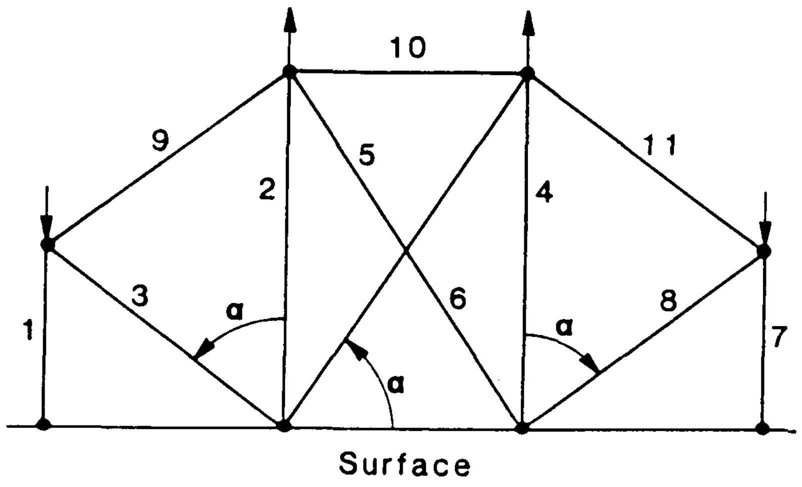

Figure 6. Initial bracing for members connecting load to support surface, OK if $\alpha \leq 75^0$

The angle, α, made by the two members is passed to the knowledge base, and a recommendation is made if the angle is not within the proper range. An example of such a recommendation is:

```
      ******RECOMMENDATIONS******
Because the angles made by the diagonals and the support surface are
greater than 75 degrees (Angle = 76.), it is recommended that members 5 and
6 be removed and that members 2 and 4 be divided in two and reconnected
with an X bracing.
Because the angle made by the diagonal and the support surface is greater
than 75 degrees (Angle = 77.5), it is recommended that member 3 be
removed and that member 2 be divided in two and connected to the ends of
member 1.
Because the angle made by the diagonal and the support surface is greater
than 75 degrees (Angle = 77.5), it is recommended that member 8 be
removed and that member 4 be divided in two and connected to the ends of
member 7.

      ************************************************
```

The truss in figure 7 reflects the refinements made from this recommendation. It is possible, especially in a very complex truss, that this recommendation might come out in addition to the recommendation about the triangles. This is another place where the user's vision will be needed to make the best choices about what members need to be removed and what members need to be added to make the best refinement to the structure.

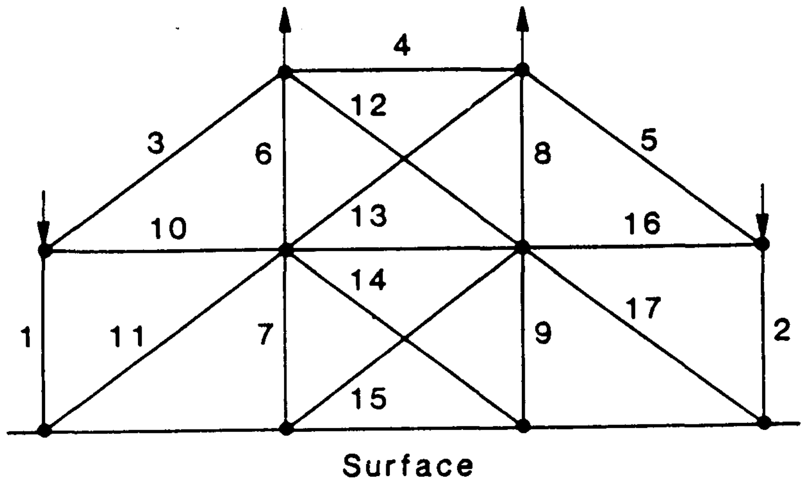

Figure 7. Bracing for members connecting loads to support surface if $\alpha > 75^0$

COMPUTER VERSUS HUMAN VISION

One of the key lessons learned in the development of this prototype system was when to rely on the computer and when to rely on the vision of the user. In the paragraphs above, there are four stages in the design process where it is seems preferable to rely on vision rather than on the computer. These include:

(1) Determining the location of the support surface relative to the loads.
(2) Connecting the load points when building a truss.
(3) Determining what members to add based on recommendations.
(4) Determining what members to remove based on recommendations.

The first attempt at determining the location of the support surface relative to the loads was done with the computer by checking X-Y coordinates of the end points of the support surface against the X-Y coordinates of the load. As long as there was a single load, this method worked well. When the capability of handling multiple loads was added, some of the loads for a single problem were found to be to the side while others were found to be above or below. Since there was no clear-cut way to resolve this problem, the easiest solution was to let the user input the location based on vision and the user's knowledge of the problem.

When building a truss, the first step is to connect the load points into a series of triangles. Determining the most appropriate triangles becomes a very complex programming problem as the number of load points increases. For a relatively small truss, it is much simpler to form the initial triangles with the aid of the user's vision, and then check those triangles for angles that are not within the required limits. The second step in building a truss is to connect all load points to the support surface without having a new member intersect an existing member. The potential intersection poses the problem within this step. For smaller trusses, it is much easier to visually make the connection between the load point and the surface and avoid the intersection, rather than check all the possible intersections and connections with the computer.

When adding members to satisfy a recommendation, it was very difficult to program the computer to do anything more than add joints to anywhere other than a known piece of information, a load point or the support surface. If a new joint for a member was needed, for example, to make a small angle larger, it would usually be placed somewhere other than a load point or the support surface. This is very complex for the computer to determine, but quite simple when relying on vision.

When removing members to satisfy a recommendation, the computer could only "see" a portion of the structure when making a recommendation. It could "see" a triangle or it could "see" a quadrilateral and make a recommendation based on that element. The complexity arises when one must determine the effects that changing one element will have on other elements. Selecting the correct members to remove (and subsequently add) is easier when the entire structure can be seen by the user.

The examples used to test STRUTEX were typically very small, less than 50 members. The numerical and symbolic techniques that would be required to handle the above problems would be difficult, but not impossible to program. Therefore, the decision was made to rely on vision to solve the problems for the time being. However, if larger, more complex structures were to be developed, it is doubtful that vision would still be the best choice in all instances. The tradeoffs need to be investigated further.

FUTURE PLANS FOR STRUTEX

Currently, there is no feedback from the analysis and optimization into STRUTEX. Plans call for the addition of a quick analysis capability which would create a file containing stress and force data about each member of the truss. A restart feature would be added to STRUTEX to allow the system to retrieve the model from the data base and plot it on the graphics window. A new rule (or rules) will be added to the knowledge base and the stress and force data will be asserted as facts to determine if there are any members under compression. A recommendation will be made for correcting the structure to check the compressed members for possible buckling.

A second rule (or rules) will be added to the knowledge base to handle feedback from the full analysis and optimization results from PROSSS. At this time the contents of this rule have not been decided.

Another interesting area that warrants attention is the simultaneous integration of CLIPS and PROLOG knowledge bases and inference engines with STRUTEX, and examine the tradeoffs between forward and backward chaining as they apply to the conceptual design problem.

Another question that remains to be answered is when do problems become large enough that it would be better to rely on the computer rather than vision. If the engineer must rely on the computer rather than vision, then what types of rules and algorithms will be required?

CONCLUSIONS

A prototype knowledge-based system has been developed to initially configure a structure to support point loads in two dimensions. There were two primary objectives for this project. The first objective was to investigate methods for passing data between a data base and a knowledge base. This was accomplished by separately integrating two types of inference engines, one forward chaining and one backward chaining, into the system and determining their effects on the flow of data between the knowledge base and the data base. No significant problems were encountered in integrating either of the inference engines. Nor did one inference engine run significantly faster than the other for this small knowledge base. It was concluded that these two systems supplement rather than compete with one another, and further research is warranted to investigate the simultaneous integration of both inference engines into the system and determine the effects on conceptual design.

The second objective was to examine when it is preferable for a computer to supply the data and when it is preferable for the data to be supplied by human vision. It was also concluded during the development of this system, that there are times to rely on the computer and there are times to rely on the vision of the user. For small problems such as the ones used for testing, there were several instances where the user's vision was more preferable than relying on the computer, such as determining the location of the support surface relative to the loads. However, for larger, more complex problems, it might be preferable to add symbolic rules to the knowledge base, numerical algorithms to the main program, and rely on the computer. More research is also needed in this area.

REFERENCES

1. Baenzinger, G.; and Arora, J. S.: Development of an Artificially Intelligent Nonlinear Optimization Expert System. AIAA Paper No. 86-0858, 1986.
2. Ceri, S.; Gottlob, G.; and Wiederhold, G.: "Interfacing Relational Databases and Prolog Efficiently." Proceedings of the First International Conference on Expert Database Systems. April 1-4, 1986, pp.141-153.
3. Chieng, W. H.; and Hoeltzel, D. A.: "An Interactive Hybrid (Symbolic-Numeric) System Approach to Near Optimal -Design of Mechanical Components." Proceedings of the 1986 ASME International Computers in Engineering Conference and Exhibition, ASME, 1986, Vol. 1, pp. 149-159.
4. Clocksin, W.; and Mellish, C.: Programming in PROLOG, Springer-Verlag, 1981.
5. DI-3000 User's Guide, Precision Visuals Inc. Document Number DI3817, Release Number 4, March 1984.
6. Erickson, W. J.: "User Guide: Relational Information Management (RIM)", Report Number D6-IPAD-70023-M, Boeing Commercial Airplane Company, Seattle Washington, 1981.
7. Feyock, S.; and Rogers, J. L.: "Embedded AI for Structural Optimization". To be presented at the International Conference on Computational Engineering Science, April 10-14, 1988, Atlanta, GA.
8. Gero, J. S.; and Balachandran, M.: Knowledge and Design Decision Processes. Working Paper. Computer Applications Research Unit, Department of Architectural Science, University of Sydney, Nov. 1985.
9. MacCallum, K. J.; Duffy, A.; and Green, S.: "An Intelligent Concept Design Assistant." Proceedings of the IFIP W.G.5.2 Working Conference on Design Theory in CAD, 1985, pp.233-249.
10. Mistree, F.; Karandikar, H.; and Kamal, S.: "Knowledge- Based Mathematical Programming: A Hybrid Approach for Decision Making in Design." Proceedings of the IFIP W.G.5.2 Working Conference on Design Theory in CAD, 1985, pp.155-188.
11. Nguyen,G.T.: "Prototypes and Database Samples for Expert Database Systems." Proceedings of the First International Conference on Expert Database Systems, April 1-4, 1986, pp.3-14.
12. Riley, G.; Culbert, C.; and Savely, R. T.: "CLIPS: An Expert System Tool for Delivery and Training." Proceedings of the Third Conference on AI for Space Applications, November 1987.
13. Rogers, J. L., Jr.; Sobieski-Sobieszczanski, J.; Bhat, R. "An Implementation of the Programming Structural Synthesis System (PROSSS)", NASA TM 83180, December 1981.
14. Rogers, J. L., Jr.; and Barthelemy, J-F. M.: "An Expert System for Choosing the Best Combination of Options in a General-Purpose Program for Automated Design." Proceedings of the 1985 International Computers in Engineering Conference and Exhibition, ASME, 1985, Vol. 2, 255-260.
15. Shah, J.: "Development of a Knowledge Base for an Expert System for

the Design of Structural Parts." Proceedings of the 1985 ASME Computers in Engineering Conference, 1985, Vol. 2, pp. 131-136.

16. Sriram, D.: A Bibliography on Knowledge-Based Expert Systems in Engineering. SIGART Newsletter, No. 89, July 1984, pp. 32-40.
17. Sriram, D.; and Joobbani, R.: Special Issue, AI in Engineering. SIGART Newsletter, No. 92, April 1985, pp. 38-144.
18. Spivey, J.: Portable PROLOG User's Guide, Department of Computer Science, University of York, Heslington, York, England, October 1983.
19. van Biema, M.; Miranker, D. P.; and Stolfo, S.J.:" The Do-Loop Considered Harmful in Production System Programming." Proceedings of the First International Conference on Expert Database Systems, April 1-4, 1986, pp.125-138.
20. Vanderplaats, G. N.: CONMIN - A FORTRAN Program for Constrained Function Minimization. User's Manual, NASA TM X_62282, 1973.
21. Whetstone, W. D.: "EISI-EAL: Engineering Analysis Language", Proceedings of the Second Conference on Computing in Civil Engineering, ASCE, 1980, PP. 276-285.

Report Documentation Page

1. Report No. NASA TM-100613	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle STRUTEX A Prototype Knowedge-Based System for Initially Configuring a Structure to Support Point Loads in Two Dimensions		5. Report Date April 1988
		6. Performing Organization Code
7. Author(s) James L. Rogers, Stefan Feyock, and Jaroslaw Sobieszczanski-Sobieski		8. Performing Organization Report No.
		10. Work Unit No. 505-63-11-01
9. Performing Organization Name and Address NASA Langley Research Center Hampton, VA 23665		11. Contract or Grant No.
		13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546		14. Sponsoring Agency Code

15. Supplementary Notes

To be presented at the Third International Conference on Applications of Artificial Intelligence in Engineering, Stanford, CA, August 8-11, 1988.
James L. Rogers and Jaroslaw Sobieszczanski-Sobieski, Langley Research Center, Hampton, VA. Stefan Feyock, VAIR, Inc., Williamsburg, VA

16. Abstract

The purpose of this research effort is to investigate the benefits that might be derived from applying artificial intelligence tools in the area of conceptual design. Therefore, the emphasis in this paper is on the artificial intelligence aspects of conceptual design rather than structural and optimization aspects. A Prototype knowledge-based system, called STRUTEX, has been developed to initially configure a structure to support point loads in two dimensions. This system combines numerical and symbolic processing by the computer with interactive problem solving aided by the vision of the user by integrating a knowledge base interrace and inference engine, a data base interface, and graphics while keeping the knowledge base and data base files separate. The system writes a file which can be input into a structural synthesis system, which combines structural analysis and optimization.

17. Key Words (Suggested by Author(s)) Knowledge base, Optimization, Conceptual design, Structural analysis	18. Distribution Statement Unclassified - Unlimited Subject Category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	22	A02