WANDERING ABOUT THE TOP OF THE ROBOT

July 1971

VISION FLASH 15

by

Patrick H. Winston

Massachusetts Institute of Technology

Artificial Intelligence Laboratory

Vision Group

ABSTRACT

Part I of this paper describes some of the new functions in the system. The discussion is seasoned here and there with parenthetical code frements that may be ignored by readers unfamiliar with PLANNER.

Part II discusses the scenario evoked in a simple sample copy effort and Part III provides some technical notes helpful to those who wish to use the system.

At the time of the first copy demonstration, the robot's command structure was still heavily pass-oriented. Lines were found, regions were parsed into bodies, and support relations were discovered in sequentially organized stages. This was done, however, with a view toward testing and proving out the modules that our more heterarchical system now provides. The newer top level organization gives us a strong foundation into which we can dump more and more information and knowledge.

Part I of this paper describes some of the new functions in the system. The discussion is seasoned here and there with parenthetical code fragments that may be ignored by readers unfamiliar with PLANNER.

Part II discusses the scenario evoked in a simple sample copy effort and Part III provides some technical notes helpful to those who wish to use the system.

I: THE FUNCTIONS

TC-COPY

As figure 1 shows, TC-COPY simply activates theorems that handle the three phases of a copying problem; namely, it calls for the spare parts to be found and put away into the spare parts warehouse area, it demands the purge of the data base to make way for information about the new scene, and it initiates the replication of the new scene.

```
(DEFPROP TC-COPY
         (THCONSE NIL
                  (COPY)
                  (Q (QUOTE (ARE YOU SURE THE RIGHT CALLIBRATION IS IN ?)))
                  ($G (PUTAWAY) (THNODB) (THUSE TC-PUTAWAY))
                  (R (QUOTE (READY TO TRY FLUSHING PICTURE)))
                  ($G (FORGETDRAWING) (THNODB) (THUSE TC-FORGET-DRAWING))
                  (R (QUOTE (DRAWING FORGOTTEN)))
                  ($G (REPLICATE) (THNODB) (THUSE TC-REPLICATE)))
         THEOREM)
```

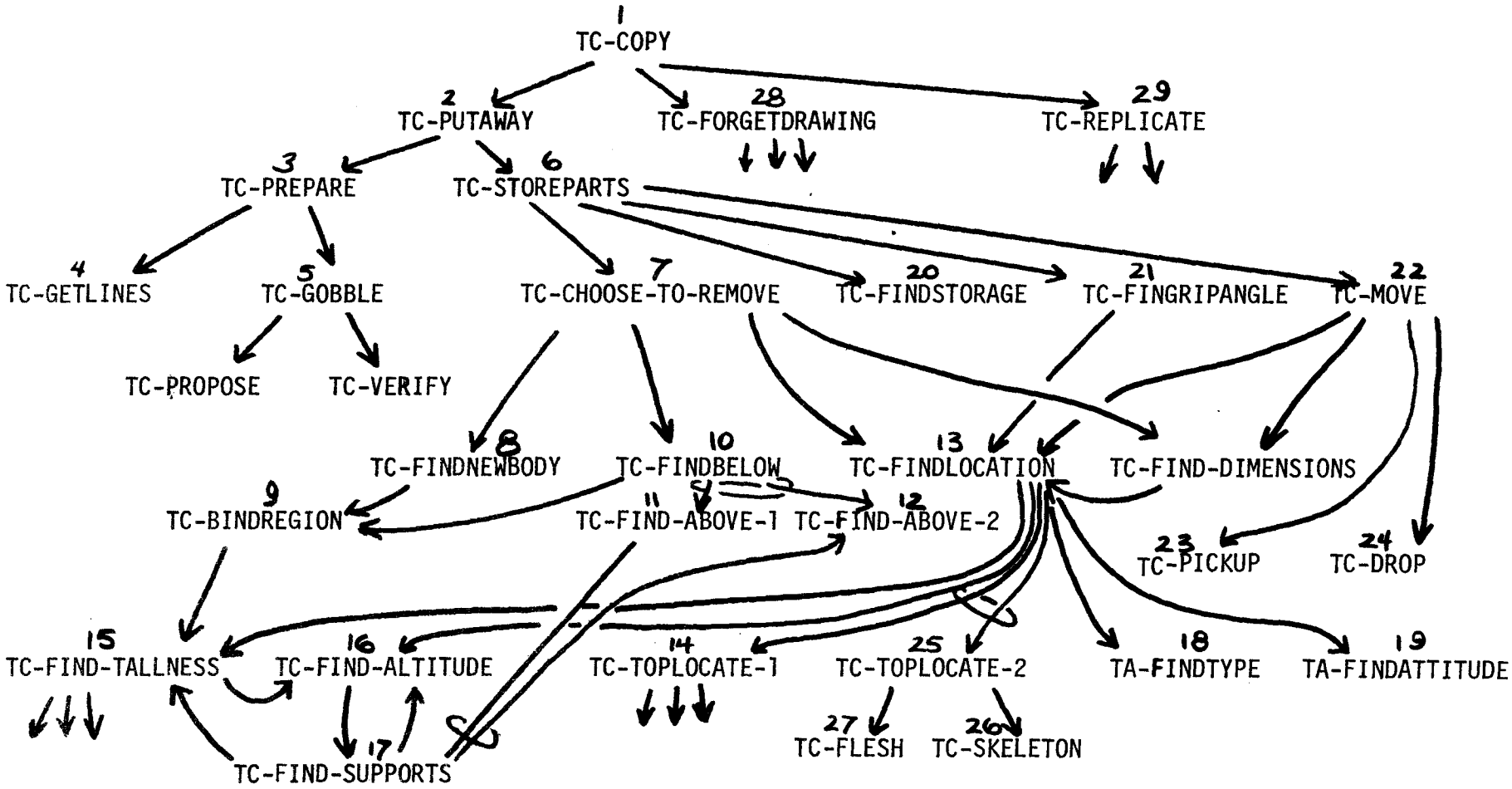


Figure 1a. The top of the system.

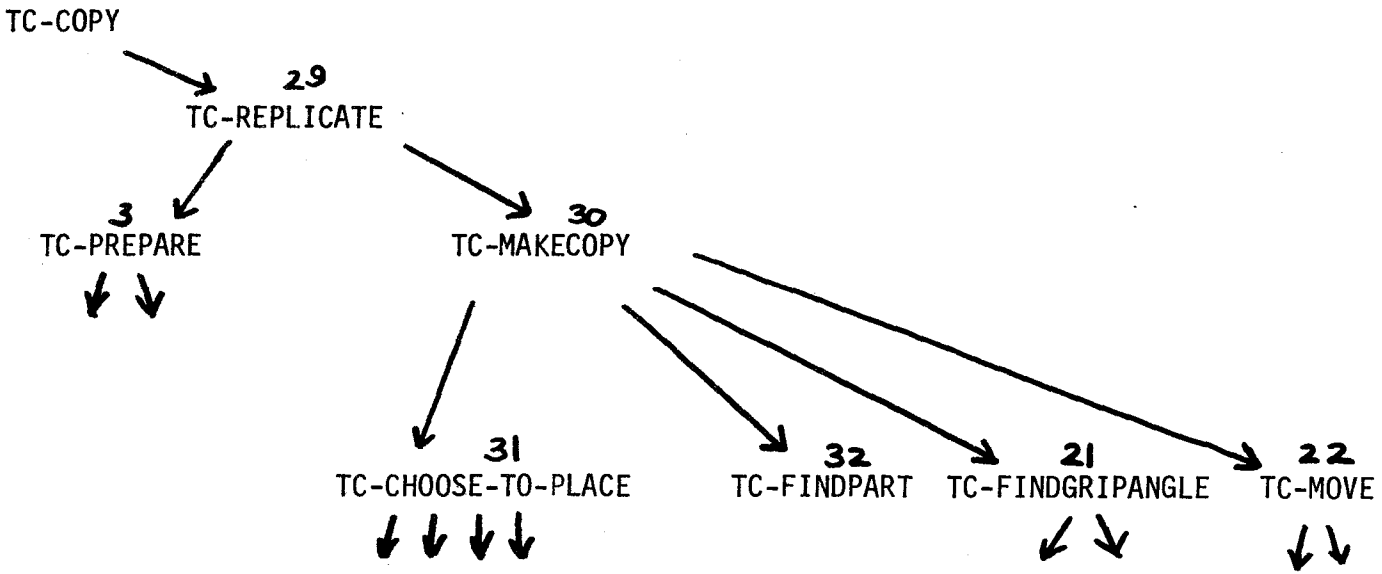


Figure 16. The top of the ystem (cont.).

TC-PUTAWAY

TC-PUTAWAY divides the problem of storing parts into two phases. During the first phase, lines are produced and bookkeeping is done. During the second, all analysis from body conglomeration upward is performed.

```
(DEFPROP TC-PUTAWAY
         (THCONSE NIL
                  (PUTAWAY)
                  (R (QUOTE (PLEASE PLACE SPARE PARTS IN FRONT OF MY EYE)))
                  ($G (PREPARE) (THNODB) (THUSE TC-PREPARE))
                  (R (QUOTE (ANALYSIS COMPLETE)))
                  ($G (MOVEPARTS) (THNODB) (THUSE TC-STOREPARTS)))
         THEOREM)
```

TC-PREPARE

The responsibilities of TC-PREPARE have been shrinking as more and more of the analysis functions are carried out on demand rather than in a pass. At the moment it first uses TC-GETLINES which handles the I/O that interfaces with the Binford-Horn-Lehrman line finder. Then it processes the hypothesized lines with TC-GOBBLE.

```
(DEFPROP TC-PREPARE
         (THCONSE NIL
                  (PREPARE)
                  ($G (GETLINES) (THNODB) (THUSE TC-GETLINES))
                  ($G (GOBBLE) (THNODB) (THUSE TC-GOBBLE)))
         THEOREM)
```

```
(DEFPROP TC-GETLINES
         (THCONSE NIL
                  (GETLINES)
                  (THDO (P (QUOTE (STARTING LINE FINDER)))
                        (BARF (QUOTE (SCAN)) EYE CALL DSK VIS)
                        (OR (AND (ERRSET SOURCE NIL)
                                 SOURCE
                                 (APPLY (QUOTE WAITFOR)
                                        (CONS (QUOTE LINES) SOURCE)))
                            (WAITFOR LINES EYE RETURN DSK VIS))
                        (P (QUOTE (LINES RECEIVED)))
                        (AND (Q (QUOTE (SHOULD I DELETE EYE RETURN FILE ?)))
                             (KILL)
                             (Q (QUOTE (SHOULD I SAVE IT UNDER LINES >)))
                             (BARF LINES LINES > DSK VIS))))
         THEOREM)
```

TC-GOBBLE

TC-GOBBLE prepares the PLANNER data base used by remaining programs.

It makes assertions of the following sort:

(R3 IS-A REGION)

(L-V1-V2 IS-A LINE)

(V1 IS-A ARROW)

(V1 IS-END-OF L-V1-V2)

(V1 LOCATED-AT 523. 627.)

(L-V1-V2 BOUNDS R3)

(ANGLE V1 V2 R3 V3 R)

The last assertion type illustrated is the most useful. As figure 2 helps show, the parts of the assertion are:

(1) the mnumonic ANGLE,

(2) the name of the vertex being described,

(3) an adjoining vertex,

(4) the region found next counter-clockwise from this adjoining vertex,

(5) the next counter-clockwise vertex,

(6) and an indicator specifying the angle as

$< 180° \rightarrow$ R

$= 180° \rightarrow$ CA

$> 180° \rightarrow$ L

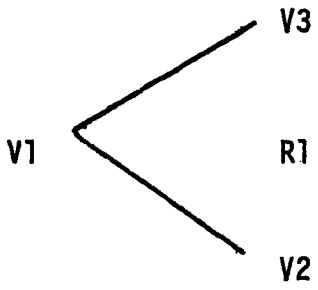


Figure 2. Elements of the ANGLE assertion

(The mnumonic for this is to imagine walking toward the central vertex from the reference spoke in this case from V2 toward V1.) If the other vertex is involved is on the right (as is V3), the indicator is R, if straight ahead CA, and if left L.

This ANGLE assertion has proven invaluable in crawling around a line drawing. In practice, a given situation may call for accessing it through a region name, a pair of vertex names, or a variety of other combinations.

TC-GOBBLE uses Eugene Freuder's TC-PROPOSE to propose potentially missing lines and to check them out with the BINFORD-LERMAN verifier.

TC-STOREPARTS

To disassemble a scene and store it, TC-STOREPARTS loops through a series of operations. It calls appropriate routines for selecting an object, finding a place for it, finding the proper angle to grip it at, and for enacting the movement to storage.

```
(DEFPROP TC-STOREPARTS
         (THCONSE (PART NEWPLACE GRIPANGLE)
                  (MOVEPARTS)
                  (LOAD (QUOTE (SKELETON PICKUP SUPPORT)))
                  LOOP
                  (THOR ($G (CHOOSE $_PART) (THNODB) (THUSE TC-CHOOSE-TO-REMOVE
                        (THAND (OR ACT
                                   (AND (P (QUOTE (STORAGE PLAN IS COMPLETE)))
                                        (Q (QUOTE (SHOULD I EXECUTE IT ?)))
                                        (PLANEVAL PLAN))
                                   T)
                               (UNLOAD (QUOTE (SKELETON PICKUP SUPPORT)))
                               (THSUCCEED THEOREM)))
                  ($G (FINDSTORAGE $_NEWPLACE) (THUSE TC-FINDSTORAGE))
                  ($G ($?PART HAS-GRIPANGLE $?GRIPANGLE)
                      (THNODB)
                      (THUSE TC-FINDGRIPANGLE))
                  (THOR ($G (MOVE $?PART $?NEWPLACE $?GRIPANGLE)
                            (THNODB)
                            (THUSE TC-MOVE))
                        (THAND (P (QUOTE (I CANNOT MOVE)) $?PART)
                               (THERT I AM WAITING)))
                  ($A ($?PART IS-A SPAREPART) (THUSE TA-REMOVE-BRICK))
                  (THGO LOOP))
          THEOREM)
```

TC-CHOOSE-TO-REMOVE

The first body examined by TC-CHOOSE-TO-REMOVE comes directly from a successful effort to amalgamate some regions into a body using TC-FINDNEWBODY. After some body is created, TC-CHOOSE-TO-REMOVE uses TC-FIND-BELOW to make sure it is not underneath something. Frequently, some of the regions surrounding a newly passed body are not yet connected to bodies, so TC-FIND-BELOW has a request link to TC-BINDREGION. (The bodies so found of course, are placed in the data base and are later selected by TC-CHOOSE-TO-REMOVE without appeal to TC-FINDNEWBODY.) If then the body under scrutiny is found to have nothing above it, TC-CHOOSE-TO-REMOVE makes a further effort to get its dimensions and position, both sets of numbers being necessary for placing the hand in the correct place.

```
(DEFPROP
  TC-CHOOSE-TO-REMOVE
  (THCONSE (B)
           (CHOOSE $?B)
           START
           (THOR ($G ($?B IS-A BODY)
                     (THNODB)
                     (THUSE TC-FINDNEWBODY)
                     (THDBF THTRUE))
                 ($G ($?B IS-A BODY))
                 (THAND ($G ($_PART IS-A BODY) (THUSE TC-FINDNEWBODY))
                        (THNOT ($G ($?PART IS-A SPAREPART)))
                        (P (QUOTE (I DID NOT GET EVERYTHING)))
                        (THOR (Q (QUOTE MAY I LOOK AGAIN ?)))
                              (THERT READY TO DEBUG))
                        ($G (FORGETDRAWING)
                            (THNODB)
                            (THUSE TC-FORGET-DRAWING))
                        ($G (FORGETBODY) (THNODB) (THUSE TC-FORGET-BODY))
                        ($G (PREPARE) (THNODB) (THUSE TC-PREPARE))
                        (THGO START)))
           (THNOT ($G ($?B IS-A SPAREPART)))
           (THNOT ($G ($?B IS-BELOW ?) (THNODB) (THUSE TC-FIND-BELOW)))
           ($G ($?B IS-AT ?) (THUSE TC-FIND-LOCATION))
           ($G ($?B HAS-DIMENSIONS ?) (THUSE TC-FIND-DIMENSIONS))
           (P $?B (QUOTE (HAS BEEN LOCATED AND MEASURED)))
           (THFINALIZE THEOREM))
  THEOREM)
```

TC-FINDNEWBODY

TC-FINDNEWBODY merely locates some unattached region and sets TC-BINDREGION to work on it. TC-BINDREGION then appeals to a collection of theorems by Eugene Freuder which do a local parse and make assertions of the form:

(R17 IS-A-FACE-OF B2)

(B2 IS-A BODY)

```
(DEFPROP TC-FINDNEWBODY
         (THCONSE (B  R)
                  ($?B IS-A BODY)
                  ($G ($?R IS-A REGION))
                  (THNOT ($G ($?R IS-A-FACE-OF ?)))
                  ($G ($?R IS-A-FACE-OF $?B) (THNODB) (THUSE TC-BINDREGION))
         THEOREM)
```

TC-FINDBELOW

As mentioned, some regions may need parsing before it makes sense to ask if a given object is below something. After assuring that an adjacent region is attached to a body, TC-FINDBELOW calls TC-FINDABOVE to do the work of determining if the body originally in question lies below the object owning that adjacent region.

TC-FINDABOVE1 and TC-FINDABOVE2

The heuristics implemented for my thesis and many of those only proposed there are now working in the TC-FINDABOVE theorems. They naturally have a bag of subordinate theormes and a link to TC-BINDREGION for use in the event an unbodied region is encountered. The assertions made are of the form:

(B3 IS-ABOVE B7)

TC-FINDLOCATION

The work of calculating location and dimension is supervised by TC-FINDLOCATION. (TC-FIND-DIMENSIONS does its job by a call to TC-FINDLOCATION.) The first task in making the calculations is to identify line-drawing coordinates of a block's top. This is done by TC-TOPLOCATE1 or TC-TOPLOCATE2. Then TC-FIND-TALLNESS and TC-FIND-ALTITUDE supply other information needed to properly supply the routine that transforms line-drawing coordinates to X Y Z coordinates. Resulting assertions are:

(B1 HAS-DIMENSIONS (2.2 3.1 1.7))

(B1 IS-AT (47.0 -17.0 5.2 .3))

Where the number lists are of the form:

(<smaller x-y plane dimension> <larger> <tallness>)

(<x coordinate> <y> <z> <angle>)

The x y z coordinates are those of the center of the bottom of the brick and the angle is that of the long x-y plane axis of the brick with respect to the x axis. Note that when TC-FINDLOCATION makes the dimension assertion, a $T results in an implicit call to two antecedent theorems by Tim Finin if they have been read in from the appropriate file. These theorems make assertions of the form:

(B12 HAS-ATTITUDE $\left\{\begin{matrix}\text{STANDING}\\\text{LYING}\end{matrix}\right\}$)

and (B12 IS-A $\left\{\begin{matrix}\text{CUBE}\\\text{BRICK}\\\text{STICK}\\\text{BOARD}\end{matrix}\right\}$)

wherever appropriate. (See Vision Flash 12.)

```
(DEFPROP TC-FIND-LOCATION
         (THCONSE (B  Z HEIGHT LOCATION DIMENSIONS TOPS)
                  ($?B IS-AT $?LOCATION)
                  ($G (TOPLOCATE $?B $_TOPS) (THNODB) (THUSE TC-TOPLOCATE1
                                                             TC-TOPLOCATE2))
                  ($G ($?B HAS-TALLNESS $?HEIGHT) (THUSE TC-FIND-TALLNESS))
                  ($G ($?B HAS-ALTITUDE $?Z) (THUSE TC-FIND-ALTITUDE))
                  (THOR (THMATCH (QUOTE ($?LOCATION $?DIMENSIONS))
                                 (FINDNUMBERS $?Z $?HEIGHT $?TOPS))
                        (THERT FINDNUMBERS FAILED))
                  ($A ($?B IS-AT $?LOCATION))
                  ($A ($?B HAS-DIMENSIONS $?DIMENSIONS) $T)
                  (THOR T (THFAIL THEOREM)))
         THEOREM)
```

```
(DEFPROP TC-FIND-DIMENSIONS
         (THCONSE (B  DIMENSIONS)
                  ($?B HAS-DIMENSIONS $?DIMENSIONS)
                  (THNOT ($G ($?B IS-AT ?)))
                  ($G ($?B IS-AT ?) (THNODB) (THUSE TC-FIND-LOCATION))
                  ($G ($?B HAS-DIMENSIONS $?DIMENSIONS))
                  (THOR T (THFAIL THEOREM)))
         THEOREM)
```

TC-TOPLOCATE1

A program by Freuder finds the four top vertices of a block if all four are visible. TC-TOPLOCATE1 and TC-TOPLOCATE2 have substantially different approaches which result in complimentary abilities.

TC-TOPLOCATE2

This program for finding the top vertexes of a brick uses TC-FLESH by Freuder and TC-SKELETON by Winston. It produces the coordinates of the actual vertexes if present, or imagined vertexes if the real ones are obscured.

TC-SKELETON

TC-SKELETON was written to cope with line finders that find only a few lines of each brick and has been adapted to its present function of identifying connected sets of 3 lines which define the dimensions of a brick. The set supplied depends on the obscuring and it is up to TC-FLESH to convert the set of 3 lines into a set of coordinates for TC-TOPLOCATE2. Figure 3 illustrates some of the 3 line skeleton sets that various flavors of obscuring cause to be generated.

TC-FIND-TALLNESS

Determining the tallness of a brick requires observation of a complete vertical line belonging to it. TC-FIND-TALLNESS uses some of TC-SKELETON's repertoire of subroutines to find a good vertical. Additionally, the altitude of the brick must be known.

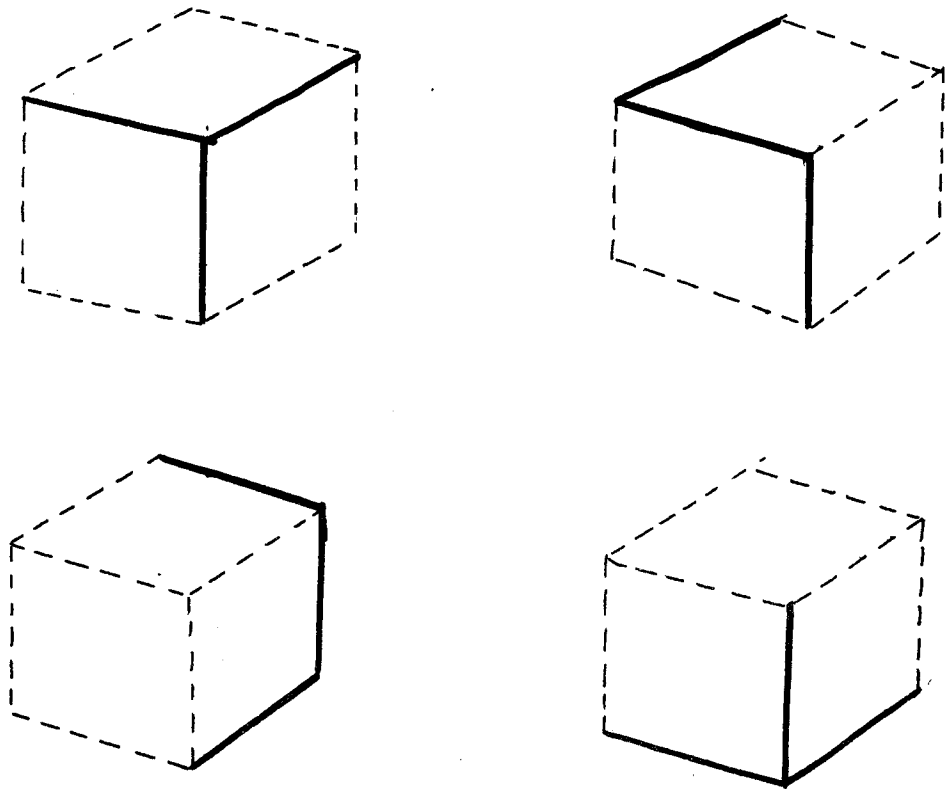


Figure 3. Some skeletons

TC-FIND-SUPPORTS

This subroutine uses TC-FIND-ABOVE to identify the objects below the one in question and then finds those objects' tallness and altitude. Adding the tallness and altitude of those objects below gives the required altitude and supports (assuming the object lies flat and so on). TC-FIND-SUPPORTS also makes an altitude assertion thus permitting TC-FIND-ALTITUDE to work by simply arranging a call to TC-FIND-SUPPORTS.

TC-FINDSTORAGE

Once an object is chosen for removal, TC-FINDSTORAGE checks the warehouse area for an appropriate place to put it. (When objects are removed from the warehouse, their bin can be reused.)

```
(DEFPROP TC-FINDSTORAGE
         (THCONSE (S CN CS XY)
                  (FINDSTORAGE $?CN)
                  (THAMONG $?XY (QUOTE ((45.0 -17.0) (40.0 -17.0)
                                                     (45.0 -11.0)
                                                     (40.0 -11.0)
                                                     (45.0 11.0)
                                                     (40.0 11.0)
                                                     (45.0 17.0)
                                                     (40.0 17.0))))
                  (THSETQ $?CN (APPEND $?XY (LIST (APPLY (QUOTE TABLEZ) $?XY)
                                                  (QUOTIENT PI 2.0))))
                  (THNOT (THAND ($G ($?S IS-A SPAREPART))
                                ($G ($?S IS-AT $?CS))
                                (EQUAL $?CN $?CS)))
                  (THOR T (THFAIL THEOREM)))
         THEOREM)
```

TC-FINDGRIPANGLE

The angle of the main axis and the dimensions of a block both enter into the computation of a proper approach by the grippers. This subroutine always chooses the angle that gives the smaller of the two possible gripper widths if the larger is above its maximum jaw opening of 2.3 inches. Otherwise it chooses the angle that puts the grippers more closely aligned with the x-axis, recognizing that our scenes tend to be spread out from left to right in front of the eye rather than from front to back and that an alignment with the x-axis therefore reduces the chance of collision with another object as the grippers come down.

```
(DEFPROP TC-FINDGRIPANGLE
         (THCONSE (LOCATION DIMENSIONS ARELATIVE A B S L)
                  ($?B HAS-GRIPANGLE $?ARELATIVE)
                  ($G ($?B IS-AT $?LOCATION) (THUSE TC-FIND-LOCATION))
                  ($G ($?B HAS-DIMENSIONS $?DIMENSIONS)
                      (THUSE TC-FIND-DIMENSIONS))
                  (THSETQ $?S
                          (CAR $?DIMENSIONS)
                          $?L
                          (CADR $?DIMENSIONS)
                          $?A
                          (CADDDR $?LOCATION)
                          $?ARELATIVE
                          0.0)
                  (THDO (THAND (GREATERP $?S 2.3000000) (THERT BLOCK TOO BIG))
                        (THAND (THOR (GREATERP $?L 2.3000000)
                                     (GREATERP (TIMES 0.75 PI) $?A (TIMES 0.25
                                                                          PI)))
                               (THSETQ $?ARELATIVE (QUOTIENT PI 2.)))
                        (THOR T (THFAIL THEOREM))))
         THEOREM)
```

TC-MOVE

To move an object to its spare parts position, the locations, and dimensions are gathered up and the gripping angle is received from the calling theorem. Then TC-PICKUP and TC-DROP interface to the machine language programs driving the arm. After TC-MOVE succeeds, TC-STOREPARTS makes an assertion of the form: (B12 IS-A SPAREPART)

```
(DEFPROP TC-MOVE
         (THCONSE (BODY DIMENSIONS
                        NEWPLACE
                        OLDPLACE
                        NEWANGLE
                        OLDANGLE
                        GRIPANGLE
                        WIDTH)
                  (MOVE $?BODY $?NEWPLACE $?GRIPANGLE)
                  ($G ($?BODY IS-AT $_OLDPLACE) (THUSE TC-FIND-LOCATION))
                  ($G ($?BODY HAS-DIMENSIONS $_DIMENSIONS)
                      (THUSE TC-FIND-DIMENSIONS))
                  (THDO (THSETQ $?WIDTH (CADR $?DIMENSIONS))
                        (THAND (GREATERP $?GRIPANGLE 0.0)
                               (THSETQ $?WIDTH (CAR $?DIMENSIONS)))
                        (THSETQ $?OLDANGLE
                                (PLUS $?GRIPANGLE (CAR (REVERSE $?OLDPLACE))))
                        (THSETQ $?NEWANGLE
                                (PLUS $?GRIPANGLE (CAR (REVERSE $?NEWPLACE))))
                        (THAND (OR (GREATERP $?NEWANGLE PI)
                                   (GREATERP $?OLDANGLE PI))
                               (THSETQ $?OLDANGLE (DIFFERENCE $?OLDANGLE
                                                              $?GRIPANGLE
                                                              $?GRIPANGLE))
                               (THSETQ $?NEWANGLE (DIFFERENCE $?NEWANGLE
                                                              $?GRIPANGLE
                                                              $?GRIPANGLE))))
                  (THGOAL (PICKUP $?BODY $E (HACKCOOR $?OLDPLACE
                                                      $?DIMENSIONS
                                                      $?OLDANGLE
                                                      $?WIDTH))
                          (THNODB)
                          (THUSE TC-PICKUP))
                  (THERASE ($?BODY IS-AT $?OLDPLACE)
                           (THUSE TE-REMOVE-ABOVES TE-REMOVE-SUPPORTS))
                  (THGOAL (DROP $?BODY $E (HACKCOOR $?NEWPLACE
                                                    $?DIMENSIONS
                                                    $?NEWANGLE
                                                    $?WIDTH))
                          (THNODB)
                          (THUSE TC-DROP))
                  ($A ($?BODY IS-AT $?NEWPLACE)))
         THEOREM)
```

```
(DEFPROP TC-PICKUP
         (THCONSE (SEND BODY)
                  (PICKUP $?BODY $?SEND)
                  (THDO (TERPRI)
                        (P (QUOTE (PICKUP AT:)))
                        (PRINTPLACE $?SEND)
                        (AND ACT
                             (BARF (CONS (QUOTE PICKUP) $?SEND) ARM CALL DSK VI
                             (WAITFOR MESSAGE ARM RETURN DSK VIS)
                             (KILL ARM RETURN DSK VIS))
                        (SETQ PLAN (REVERSE (CONS (CONS (QUOTE PICKUP) $?SEND)
                                                  (REVERSE PLAN))))))
         THEOREM)
```

TC-FORGETDRAWING

Before the scene to be copied may be analyzed, the old drawing must be purged. TC-FORGETDRAWING does this through three theorems, TA-FORGET-LINES, TA-FORGET-VERTEXES, and TA-FORGET-REGIONS.

TC-REPLICATE

To make the copy, TC-REPLICATE performs much as TC-PUTAWAY, but with TC-MAKECOPY, TC-CHOOSE-TO-PLACE, and TC-FINDPART replacing TC-STOREPARTS, TC-CHOOSE-TO-REMOVE and TC-FINDSTORAGE. Assertions of the form:

(B12 IS-A SPAREPART)

(B2 IS-A-PART-OF COPY)

(B2 IS-ABOVE B1)

are kept up to date throughout by appropriate antecedant theorems.

TC-CHOOSE-TO-PLACE

Objects are placed after it is insured that their supports are already placed.

TC-FINDPART

The part to be used from the warehouse is selected so as to minimize the difference in dimensions of the matched objects. Specifically, one searched until:

$$\sum_{\text{dimensions}} \left| \langle \text{spare part dimension} \rangle - \langle \text{model part dimension} \rangle \right|$$

is less than a sliding threshold.

II: A SCENARIO

To illustrate how the robot functions, I use the following conventions, together with the diagram of figure 4:

(1) Theorems are represented by the numbers given in figure 1.

(2) The history of theorem calls proceeds from left to right.

(3) Theorems called are placed directly under the calling theorem.

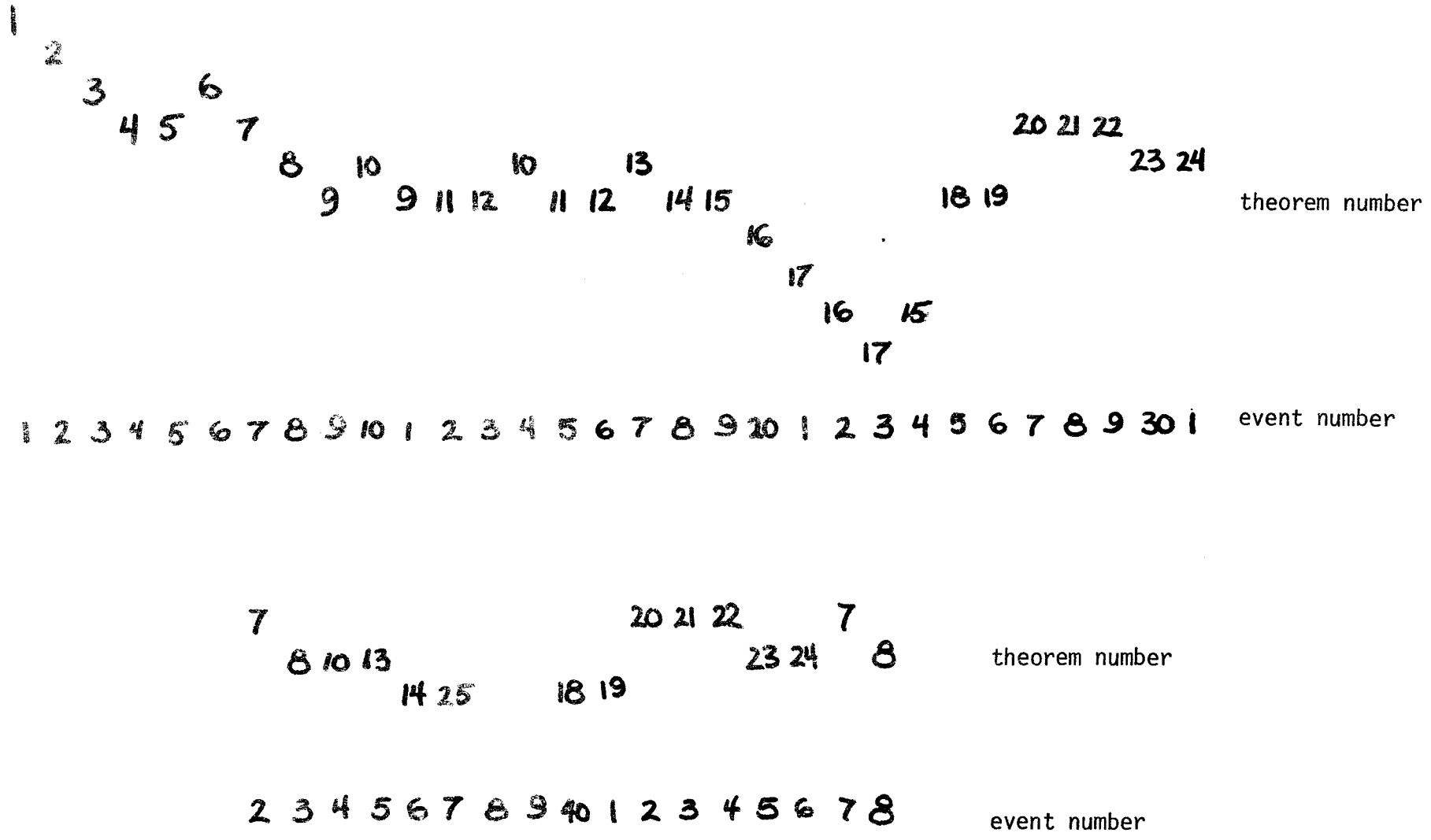
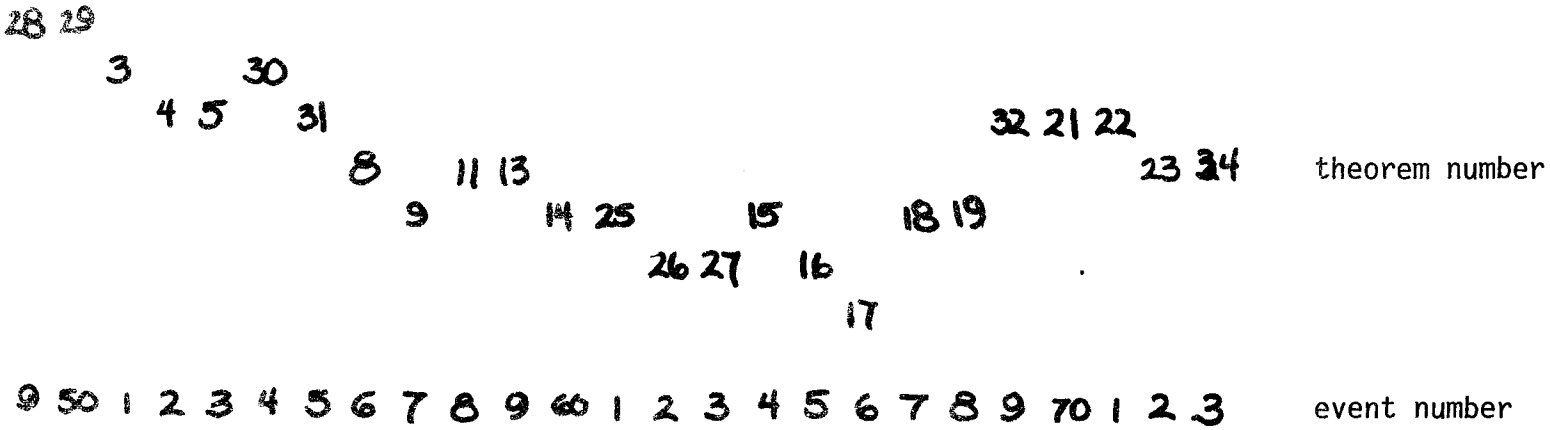


Figure 4. A sample trace.



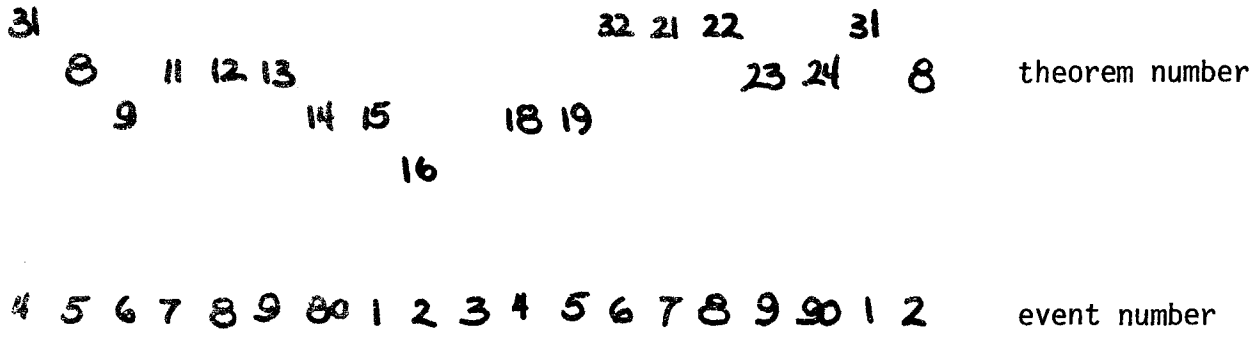


Figure 4. A sample trace (cont.).

In what follows, the scenes are those of figure 5. Identical, simple scenes without missing lines were chosen for ease of tracing through the systems operation.

Events 1-5

TC-COPY begins the activities that carry through eventually to event 92. For the scenes here, nothing interesting happens during the first few events as no lines were missed by the line finder.

Event 6

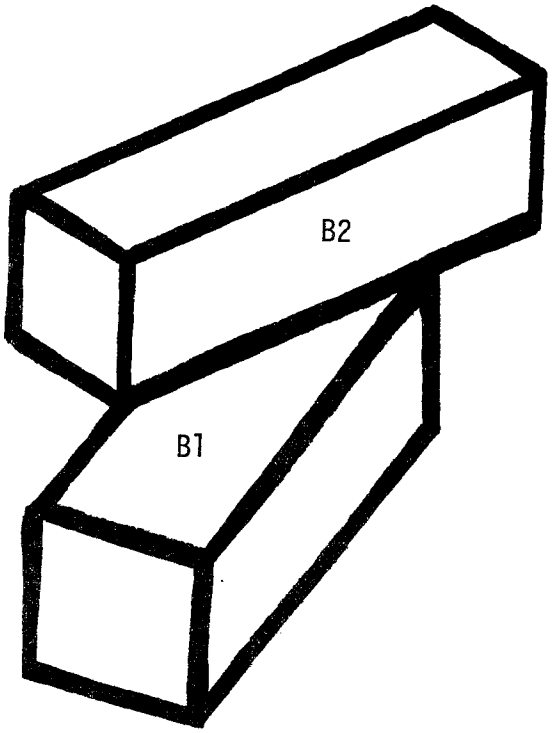
TC-STOREPARTS begins supervision of disassembly. Note the two 7-20-21-22 sequences on the line below. The 7's are widely separated from the 20's because of the many operations required to choose a brick for removal.

Events 7-9

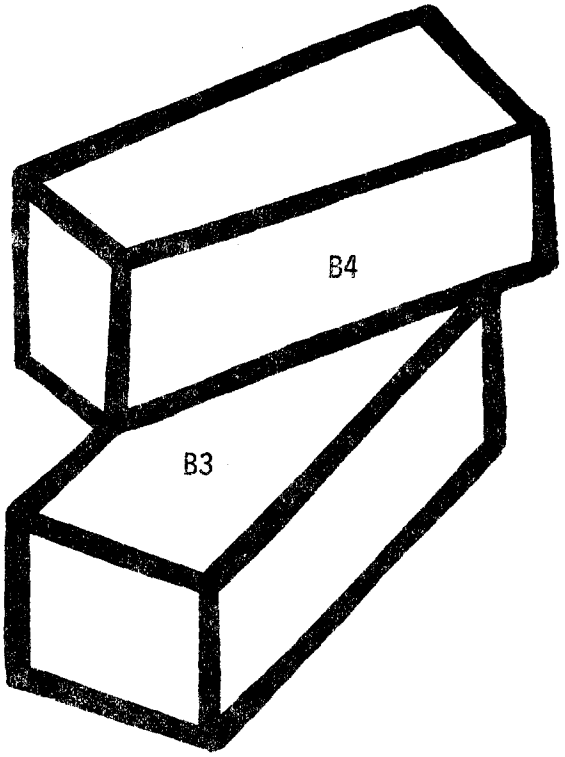
TC-CHOOSE-TO-REMOVE parses a few regions together into a body, B1.

Events 10-13

A check is made to insure that the body is not below anything. Note that B2 is parsed during this phase as required for the TC-FIND-ABOVE routines. Unfortunately B1 is below B2 and therefore TC-CHOOSE-TO-REMOVE must select an alternative for removal.



Objects to be used as parts



Scene to be copied

Figure 5.

Events 14-16

B2 was found while checking out B1. TC-CHOOSE-TO-REMOVE now notices it in the data base and confirms that it is not below anything.

Events 17-24

The top vertexes of B2 are obtained through TC-TOPLOCATE-1 which works well on such highly exposed bricks. TC-FINDLOCATION then proceeds to nail down other location and dimension parameters for B2. As indicated by the depth of call, this requires something of a detour as one must first know B2's altitude, which in turn requires some facts about B1. Note that no calls are made to TC-FIND-ABOVE routines during this sequence as those theorems previously were used on both B1 and B2 in determining their suitability for removal.

Events 25-26

Tim Finin's theorems establish that B2 is a lying brick, given knowledge about the block's dimensions.

Events 27-31

Space in the warehouse is found, a gripper angle is determined, and the object is moved to storage.

Events 32-34

TC-CHOOSE-TO-REMOVE begins looking for another object. TC-FINDNEWBODY fails as all bodies have been parsed, but B1 is picked up from the data

base. TC-FIND-BELOW notes that the only object above B1 has become a spare part.

Events 35-39

TC-FINDLOCATION fails in an attempt to use TC-TOPLOCATE-1 on B1 because it is badly obscured. TC-TOPLOCATE-2 works fine, however, through calls to TC-FLESH and TC-SKELETON. There is no hint of the complicated 15-16-17-16-17-15 altitude determining sequence because the necessary observations about B1 were made in the course of locating B2.

Events 40-41

B1 is a lying brick.

Events 42-46

B1 is put away.

Events 47-48

TC-CHOOSE-TO-REMOVE fails to find anything else to move and therefore success propagates to TC-COPY.

Event 49

The line drawing is purged.

Events 50-57

Brick B3 is parsed.

Event 58

TC-CHOOSE-TO-PLACE uses TC-FIND-ABOVE-1 to determine that B3 is on the table and that a matching object may be placed in the copy immediately.

Events 59-68

B3 is located and measured.

Events 69-73

A matching spare part is found in the warehouse and moved into position as the first block in the copy.

Events 74-76

A search is made for another object. B4 is parsed here, as that parsing was not required to determine the suitability of placing an object for B3.

Events 77-78

TC-FIND-ABOVE-1 fails but TC-FIND-ABOVE-2 determines that B4 is supported by something already replicates in the copy.

Events 79-90

Dimensions and positions are determined, a mate is found, and the mate placed.

Events 91-92

No more parts are found to be copied into the copy, so TC-COPY succeeds.

III: TECHNICAL NOTES

The robot is run from a special MICRO-PLANNER that differs from the standard one only in that it is larger and comes equiped with some files already loaded. To get it, incant as follows to DDT:

PLNR$U

$L DSK:VIS;TS PLNR

as soon as the system responds with a *, type $G and wait for whatever happens to be the current announcement of the top-level THVAL loop to be printed.

Then type:

(READY)

Then typing:

(THVAL GA)

does everything, as the value of GA is

(THGOAL (COPY) (THNODB)(THUSE TC-COPY)).

To walk through in smaller steps, with time to reflect and protect progress so far, the user types

(THVAL Gx)

The value of GB, for example, is (THGOAL (PREPARE) (THNODB) (THUSE TC-PREPARE)). The values of other current Gx's can be obtained by typing (GS).

Should the user wish to store the data base at some point anticipating

bugs later on in the analyses, he types (SAVE) which writes into DSK: VIS; SAVE >. Then (RESTORE) cleans out the data base and brings everything back from SAVE >.

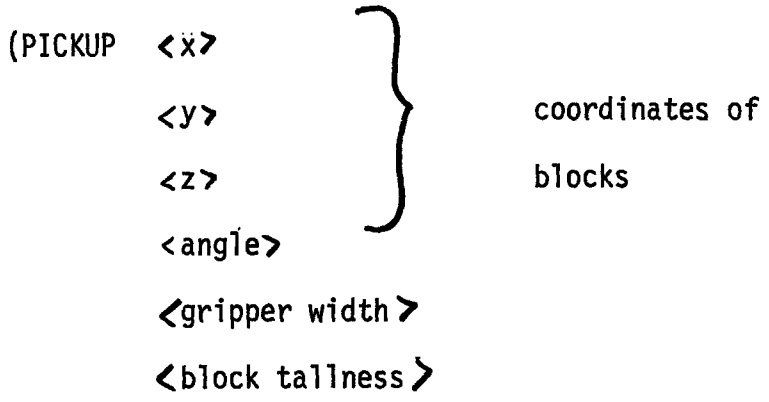
In the course of analysis, the PLANNER based operations communicate with other programs via disk files or the core link devise. A set of input lines is a list of lists of four numbers X1 Y1 X2 Y2. For example, a rectangle would come over as:

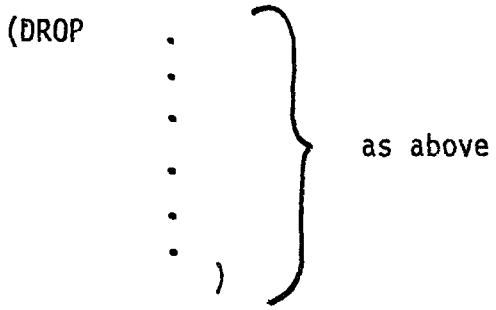
```
((400   400   400   500)
 (400   500   500   500)
 (500   500   500   400)
 (500   400   400   400))
```

As suggested, the numbers typically range over the midground between 0 and 1,000. They are decimal, fixed point numbers. They are found by TC-GETLINES in the file EYE RETURN in the VIS disk directory. Requests for line verification are answered through a new file of the same name. (The files are deleted just after reading.)

Eye action is initiated through a file written on the disk with the name EYE CALL. Its contents determine the eye action taken. Currently, these contents might be (SCAN) or (VERIFY <x1> <y1> <x2> <y2>).

Communication with the arm is handled in a similar way, with the file ARM CALL, which initiates action, and ARM RETURN, which reports on the success or failure of the action, insofar as the arm itself can tell. The arm calls are currently on the form of files with one of the following in them:





The machine language arm program itself has some ability to cope with variability. If one of the micro-switches mounted on the ends of its two fingers is activated too soon, it moves over a bit and tries again. If a micro-switch in the wrist is activated indicating that an attempt is being made to push an object down through another, it backs off and tries again with a modified altitude parameter. This is all in the wrong place and will be moved over to PLANNER code eventually.