PROTOTYPE SPATIAL REASONING PROJECT

FINAL REPORT

SAE-DC-86-R-042

January 22, 1986

For

Research Institute Center for Artificial Intelligence
Engineer Topographic Laboratory
U.S. Army
Ft. Belvoir, Virginia

By:

Software Architecture and Engineering, Inc.
1500 Wilson Boulevard, Suite 800
Arlington, Virginia 22209
DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.
A demonstration prototype system based on expert system technology in which both diagnostic and spatial reasoning techniques can be brought to bear on a problem. The system design allows for the logically different types of reasoning to cooperatively solve classes of problems that have non-spatial and spatial aspects.
Prototype Spatial Reasoning Project
Final Report

Barry T. Perricone
Jane Potts

January 22, 1986

Work Performed By: Software Architecture & Engineering, Inc.
SAE Job Control : 110592

For: Research Institute Center for Artificial Intelligence
Engineer Topographic Laboratory
U.S. Army
Preface

This report represents the last deliverable under CONTRACT NUMBER N00014-82-C-0428/P00008.
1. Introduction

1.1. Project Objectives Review

The main objective of the spatial reasoning project, briefly stated, is to:

Demonstrate the feasibility of using expert system technology as an aid to tactical mission planning and in particular to demonstrate the use of spatial reasoning in this context through a prototype system capable of knowledge-based spatial deduction.

The prototype demonstration addresses a subproblem of tactical mission planning, the determination of likely locations for enemy artillery batteries. This determination is based on various types of supplied and inferred information. The supplied information consists of a symbolic description of the 3-D relationships that are known to exist between the components of an artillery formation (e.g., three artillery batteries not more than 20 meters apart), and other known placement constraints based on the properties of the 3-D object being modeled (e.g., an artillery battery cannot be positioned in a geographic area that contains more than 1 meter of water).

The scope of the enemy artillery battery placement determination is, in the demonstration system, restricted to one possible artillery battery formation and four constraints:

The formation consists of three distinct artillery batteries that must be in a straight line. The distance between a battery and its immediate neighbor battery can be 2-4 meters inclusive (3 being the optimum) along the X axis, and a 0-5 meter difference between battery neighbors (0 being the optimum) along the Z axis.

The absolute difference in elevation between any member of a formation cannot exceed 15 meters.

The distance between the leftmost artillery battery and the rightmost artillery battery contained in the formation must be between 3-24 meters inclusive.

A line of sight must exist between neighboring artillery batteries contained
within the same formation.

The "line" formation must be parallel to the FEBA (Front Edge of the Battle Area) that is defined interactively by the end-user of the demonstration system.

This problem is sufficiently rich to demonstrate the potential of expert system technology and the use of spatial reasoning. (For a more detailed discussion of the problem see "A Prototype Demonstration of Spatial Reasoning as Applied to Army Tactical Planning Problems" SAE document number SAE-DC-83-P-013.)

1.2. Report Organization

The remainder of this report is divided into the following sections:

Status of Tasks - presents a brief account of what has been accomplished relative to the proposed tasks of the project.

Conclusions - presents the general results of the prototype effort and discusses the insights gained into the problem area and recommendations for further work.

Appendices - a collection of information that consists of a description of the deliverables and their location within ETL's VAX 780 file system, instructions on how to invoke and interact with the demonstration system developed, and instructions on how to invoke and interact with a "slide show" of formation placements generated previously through the demonstration system. Also included as an addendum is the BNF developed for the spatial reasoning system.
2. Status of Tasks

There are a number of tasks that were covered under the contract relative to Phase 1 and 2 of the project. Phase 1 tasks were completed and demonstrated to the client at the completion of Phase 1. A discussion of Phase 1 tasks will, therefore, not be addressed here. Phase 2 tasks were as follows:

Complete the demonstration prototype.

Complete spatial expert system, include justification capabilities and domain independent spatial knowledge base parser.

Expand demonstration system to handle multisegment FEBA, sector of interest, side, and bands.

Unfortunately, some of the Phase 2 tasks listed above were not done or totally completed due to insufficient funds. At the beginning of Phase 2 we anticipated that this might happen so we choose to put our effort into those tasks that would still allow a proof of concept for the thesis of the demonstration system, i.e., spatial reasoning. What follows is a discussion of how well we accomplished each of the Phase 2 tasks listed above.

2.1. Demonstration Prototype

The demonstration system prototype is complete. The source code for its various functional components is resident on ETL's VAX 780 file system. The file organization of the source code in an annotated format is presented in Appendix A. The demonstration system is fully functional and can be executed. Instructions on how to run the demonstration system and interact with it is given in Appendix B and C.

2.2. Spatial Expert System

The foundation for the restricted spatial expert system shell proposed is complete and functional. However, due to insufficient funds all of the enhancements to it could not be completed fully and are therefore not operational. All of the code that was produced relative to these enhancements are present on ETL's VAX 780. An enumeration of these enhancements and their present status is presented next.

2.2.1. Justification Capabilities

A very rudimentary justification capability for the spatial reasoner is coded, however, it has not been
integrated into the spatial reasoning system. It would require more work to functionally enhance it so as to be useful and integrate it. Therefore, no justification capabilities relative to the spatial reasoner exist within the demonstration system.

2.2.2. Knowledge Base Parser

A large amount of work went into the development of a domain-independent parser for the spatial reasoner. Both the lexical analyzer and parser are very close to completion. Minor alterations to the lexical analyzer and moderate changes to the parser are required. Also required to complete the knowledge base parser are changes to the internal representational format of the spatial information within the spatial reasoning system. Without these changes the knowledge base parser is only partially operational. The BNF for the syntax developed for the knowledge base parser is given as an addendum to this report.

Since there is no parser available for the spatial reasoning system the information that would normally be contained within it is part of the initialization code of the spatial reasoning code proper. Therefore, in order to change the "knowledge base" used, the initialization code for the spatial reasoner must be changed directly. This code basically performs the same tasks that the parser would perform as side effects. It is important to note that the spatial reasoning system is domain independent and the fact that the "knowledge base" has to be entered in this unusual manner does not alter this property.

2.3. Expansion of Demonstration System

After completing the work described very little time on the contract remained. None of the demonstration system expansions were performed: multisegment FEBA, sector of interest, side, and bands. However, none of these enhancements are needed to provide the proof of concept sought after.
3. Conclusions

The demonstration system performs its intended task, the placement of artillery battery formations based on spatial constraints and the properties of physical objects. It accomplishes this task by using prototypic spatial inferencing techniques developed by Software A&E personnel.

The correct functioning of this demonstration system represents the proof of concept desired. It has been demonstrated (and therefore proven) that it is possible to infer spatial information by the use of computer-based expert systems.

There are several avenues of research that this prototypic system opens. First, the system is slow. It takes approximately 10-20 minutes to infer placements within a 120 meter squared area. This time can possibly be shortened by developing a more efficient internal representation of the spatial model being worked upon. More research needs to be done in this area. Secondly, the spatial reasoner can be made more powerful by embedding "deep" spatial knowledge within it. A more sophisticated graphically display could be researched and implemented thereby increasing the ease of comprehension for the end-users of such a system.

In conclusion, the fact that it is slow and could be enhanced does not take away from the realization that computer-based spatial reasoning is possible. We have proven it through the existence of the demonstration system developed through this contract.
APPENDIX A

DELIVERABLES ROADMAP
Appendix A: Deliverables Roadmap

The following is a roadmap to the contract deliverables. Names that are underlined represent directories, names indented under these directories are the files/directories contained within them.

DELIVER

demo-run
  executable for a demonstration run of the spatial reasoning system
feba_data
  temporary file created by demo (gs) - contains endpoints of feba
filename.tap
  temporary file created by demo (gs) - contains name of image file
gs
  graphics executable called by spatial reasoning system
gs-parameters
  temporary file created by demo (ses)
gscoords
  temporary file created by demo (gs) - coordinates of center point of area of interest
positions_data
  temporary file created by demo (ses) - artillery placements determined by the spatial reasoning system
primary.elev
  temporary file created by demo (gs) - image file of elevation for display on the grinnell created from the raw data of the catts data
primary.hydro
  temporary file created by demo (gs) - image file of hydrography for display on the grinnell created from the raw data of the catts data
r_positions
  temporary file created by demo (ses) - ?
ses
  executable of the spatial reasoning system
spatial-demo.kb
  the KES.PS knowledge base that controls the operation of the demonstration system
subarea_data
  temporary file created by demo (gs) - contains coordinates, elevation, and hydrography of each pixel in the subarea of interest

COMMON:

directory contains the source code common to both the Shared Information System and original Graphics system. These source files have been replaced with an enhanced version of the system. They are no longer needed and are present solely to give the client all the code developed under the contract.
KES:

directory contains enhancements to KES 1.4.3 for use with the spatial reasoning system

Ecntrl.l
control functions
Econclude.l
conclude functions
Eexternal.l
external functions
Egetargs.l
get arguments functions
Ehelp.l
help functions
Enassert.l
assert functions (cassert implemented)
Enstop.l
stop functions (sx implemented)
Eps.o
relocatable code of modified kes.ps
Estatus.l
status functions
Estmt.l
statement functions
Makefile
Makefile to generate Eps.o and modified kes.ps
modified-kes.ps
executable of modified kes.ps

NEW-GRAPHICS:

directory contains the source code and executable of the enhanced graphics system that is called by the spatial reasoning system

Makegraphics
Makefile to create gs, the graphics executable used by the spatial reasoning system
getcatval.c
c file included in gfuns.c which gets values from the catts raw data
gfuns.c
c source file for gs, the graphics system. Includes cursor routines written specifically for the project
gfuns.c,v
archived (rcs) gfuns.c
gpconstants.h
included by gfuns.c, contains constant graphics declarations
gs executable of the graphics program called by the spatial reasoning system

NEW-GRAPHICS/TESTING:

subdirectory for running tests of the graphics system

feba_data  temporary file created by gs - contains endpoints of feba
filename.tmp temporary file created by gs - contains name of image file
gs-parameters temporary file created by ses
gscoords temporary file created by gs - coordinates of center point of area of interest
positions_data temporary file created by ses - artillary placements determined by the spatial reasoning system
primary.elev temporary file created by gs - image file of elevation for display on the grinnell created from the raw data of the catts data
primary.hydro temporary file created by gs - image file of hydrography for display on the grinnell created from the raw data of the catts data
subarea_data temporary file created by gs - contains coordinates, elevation, and hydrography of each pixel in the subarea of interest

OLD-GRAPHICS:

old graphics code for earlier version

GSgetcatimg.c included in gpfcns.c, get raw catts data
Makefile make test relocatable, gtest.o
READ.ME readme file
gchar.c included in gscaller.c, get character routine
gchar.h included in gchar.c, get character definitions
ggetcatval.c included in gpfcns.c, c file which gets values from the catts raw data
gpconstants.h included in gpfcns.c, contains constant graphics declarations
gpfcns.c c file to make old graphics system
gpfcns.h included in gpfcns.c, contains graphics functions
graphics.doc
documentation of old graphics
gscaller.c
  c source file of calling routine of old graphics system
gscaller.o
  relocatable of calling routine of old graphics system
gtest.c
  graphics test source code file
int_gs.l
  lisp graphics initializer and loader source file
int_gs.o
  lisp graphics initializer and loader relocatable
shading.c
  included in gpfcns.c, c source file for shading
test.c
  test graphics system
test2.c
  test graphics system

NEW-SES:

---

new spatial reasoning system

ETL-main.l
  lisp source code main caller file
ETL-main.o
  lisp relocatable main caller file
README
  readme for new spatial system
SES
  executable of new spatial system
angles.l
  lisp source files concerning angles
angles.o
  lisp relocatable files concerning angles
begin.l
  lisp source files to create the standalone
  spatial reasoning system used in the
demonstration system.
commands.l
  lisp source files concerning commands
commands.o
  lisp relocatable files concerning commands
compile.l
  lisp source files concerning compilation
cstack.l
  lisp source files concerning command stack
cstack.o
  lisp relocatable files concerning command stack
ext_file.l
  lisp source files concerning externals
ext_file.o
  lisp relocatable files concerning externals
format.l
lisp source files concerning format
  globals.l
  globals.o
lisp relocatable files concerning globals
  info_space.l
  info_space.o
lisp source files concerning globals
  info_space.o
lisp relocatable files concerning information space of system
parse.l
lisp source files concerning command parser of system
parse.o
lisp relocatable files concerning command parser of system
sys.l
lisp source files concerning system
sys.o
lisp relocatable files concerning system
wkspace.l
lisp source files concerning work space of system
wkspace.o
lisp relocatable files concerning work space of system

NEW-SES/PARSER:
files and directories necessary to create the parser, and to invoke
lisp functions acting upon parsed objects

Makeparser
Makefile to make the parser invokes yacc and lex as well as the
c compiler, plus some special utilities necessary to the interface
between lisp and c
callparse.l
lisp file that calls the c relocatable of the parser
justify.l
lisp functions for justification of placement of objects by the
spatial system
main.l
lisp source main for sample parser
main.o
lisp relocatable main for sample parser
model.l
lisp functions for initial model instantiation
spatial-kbl
sample kb for the parser
startfuns.l
lisp startup functions loaded before calling the parser
startup.l
startup file to load lisp files and the parser
template.l
lisp functions for templates (used by justification and
initial models)

PARSER/data:

-10-
test data files to test the different sections of the parser created for the spatial system - must be concatenated together to create one data file:

    cat con.dat prim.dat obj.dat init.dat com.dat > test.dat

com.dat
commands section
con.dat
constants section
init.dat
initial models section
ll.dat
sample test data file
obj.dat
application objects section
prim.dat
user primitives section

PARSER/defs:

-------------
definition file(s) for the parser

y.tab.h
definition file for the parser and lexical analyzer generated by yacc

PARSER/doc:

------------
documentation concerning lisp and the lisp-c interface

Lisp_C.doc
interface between lisp and c, written by J. K. Potts
franz.doc
description of lisp on the vax by John Foderaro

PARSER/interm:

------------
intermediate c files generated by lex and yacc

lex.yy.c
c file generated by lex for lexical analyzer of the spatial system

y.tab.c
c file generated by yacc for parser of the spatial system

PARSER/reloc:

------------
relocatables for lexical analyzer and parser

lex.yy.o
relocatable for lexical analyzer, generated by lex
spatial.o
relocatable for spatial system parser, lexical analyzer
and parser linked together and called from lisp

y.tab.o
relocatable for parser, generated by yacc

session.dat
sample session of the parser called from lisp

PARSER/source:
-------------
source code for parser and lexical analyzer

lxspatial.c
C source file for lexical analyzer - input to lex

lxspatial.c,v
archived (rcs) lxspatial.c

objclass.l
lisp functions for displaying object classes

savesp.y
earlier version of c source file for parser - input to yacc

spatial.y
C source file for parser - input to yacc

spatial.y,v
archived (rcs) spatial.y

NEW-SES/TESTING:
-----------------
contains versions of ETL-main.l that were
used in testing of the demonstration system
along with needed data files.

ETL.l
ETL2.1
feba_data
fulda.gen
positions_data
r_positions
subarea_data

OLD-KBS:
-------
knowledge bases that drove the old spatial system

des.kb
decision supervisory system

ipses.kb
interface protocol system

OLD-KBS/testkbs:
----------------
test knowledge bases to ascertain the
correctness of the modified KES/PS system
developed for this contract
OLD-SIS:

directory contains the source code for the
Shared Information System. These source
files have been replaced with an enhanced
version of the system. They are no longer needed and
are present solely to give the client all the code
developed under the contract.

Makefile
sisex.c

SLIDE-SHOW:

sample sessions of the spatial system for demonstration.
Includes the saved data files from previous sessions
so that they can be presented during the execution of slide-show.

x_feba_data
x_filename.tmp
x_gscoords
APPENDIX B

HOW TO USE THE DEMO
Appendix B: How to Use the Demo

Very simple. Change your directory so that your present working directory is "/etl/other/barryp/deliver":

```
cd /etl/other/barryp/deliver
```

Enter the command "demo-run" and then follow the directions presented to you on the screen. The system is very easy to use and there is a tutorial built into the system. The opportunity to view this tutorial will be offered to you as a choice to the first system generated question to you.
APPENDIX C

HOW TO USE THE SLIDE SHOW
Appendix C: How to Use the SLIDE SHOW

Very simple. Change your directory so that your present working directory is "/etl/other/barryp/deliver":

    cd /etl/other/barryp/deliver

Enter the command "slide-run" and then follow the directions presented to you on the screen.
APPENDIX D

GRAPHICS MODULE
Appendix D: Graphics Module

GS - The Graphics Module

FUNCTION

This module provides the use of the graphic capabilities of the grinnell to the spatial system. A catts raw data set for a 512 by 512 image is processed to produce two image files: one for elevation of the area; one for the hydrography of the area. The user is asked to define a FLEA (forward edge of the battle field) and a subarea of interest (the latter is an 11 X 11 pixel square). The results of the spatial reasoning system may be displayed on the grinnell.

The gs program is called by the spatial system in one of three modes:

image: creates the image file for display on the grinnell from the raw catts data, then prompts the user to designate the FLEA and the area of interest
reuse: uses the existing image file, created by an earlier call to gs, and prompts the user to designate the FLEA and the area of interest
placements: displays placements determined by the spatial reasoning system

PARAMETERS

The valid parameters to the gs program are:

\texttt{gs image \langle catts raw data\rangle \langle input parameters file\rangle}
\texttt{gs reuse}
\texttt{gs placements \langle name of placements file\rangle}

INPUT FILES

The files needed by the graphics module are (\langle\rangle indicates command line parameters):

\texttt{for image:}
\texttt{\langle /imgg/catts/fulda/raudatm/fulda.SIZ\rangle - raw catts data}
\texttt{\langle gs-parameters\rangle - x and y coordinates of lower left corner}
\texttt{for reuse:}
\texttt{primary.elev - image file of elevation}
\texttt{primary.hydro - image file of hydrography}
\texttt{for placements:}
\texttt{primary.elev - image file of elevation}
\texttt{primary.hydro - image file of hydrography}
<positions_data> - 3-point coordinate data
eba_data - feba endpoints
gscoords - coordinates of center of area of interest

OUTPUT FILES

The files generated by the graphics system are:

by image:
  primary.elev - image file of elevation
  primary.hydro - image file of hydrography
  feba_data - feba endpoints
  gcoords - coordinates of center of area of interest
  subarea_data - coordinates, elevation and hydrography of
                  points in subarea of interest
  filename.tmp - file containing name of image file created

by reuse:
  feba_data - feba endpoints
  gcoords - coordinates of center of area of interest
  subarea_data - coordinates, elevation and hydrography of
                  points in subarea of interest
  filename.tmp - file containing name of image file created

by placements:
  none

COMPILATION

A Makefile called Makegraphics will provide all the necessary linking
to be done to produce the executable:

@ Home directories.
ROOT = /isu/tb

@ Library archives.
LIB = $(ROOT)/LIB
VLSELIB = $(LIB)/visionlib.a
CMULIB = $(LIB)/cmulib.a
SUBLIB = $(LIB)/sublib.a
IMGLIB = $(LIB)/imagelib.a

gs: gfunsc.c
    cc -g gfunsc.c $(VLSELIB) $(CMULIB) $(IMGLIB) $(SUBLIB) -lm
    mv a.out gs

FUNCTIONS

A brief description of the internal functions of the graphics module follows:

GSgetcatimg - Read raw catts data file and convert to image file
box_point  draws red overlay box around x and y point with side length of len, and returns 2 sets of x and y coordinates that define the box

check_coords  checks that feba points are on valid sides

create_img_file  Create image file from raw catts data — calls GSgetcating

define_feba  Routine for defining feba

display_results  displays results of spatial system in right hand corner of grinnell

display_x_y  updates display of x, y and step values of grinnel cursor in lower right corner of grinnel screen

error_usage  Prints error message about usage of go program

expand  expands the area of source frame (img) defined by x1, y1, x2, y2 and writes to upper right hand corner of destination frame

get_kcur  Keyboard cursor routine

get_cursor  gets cursor position maxpoints times, storing x and y positions in xarray and yarray

g_feba  Put up image frame and obtain feba

main  determines mode (image, reuse, display) and call appropriate subroutines

num_to_string  converts integer to string

print_file_coords  Prints xyz coordinates in Kes format

raw_read  reads one raw input character from keyboard — does not echo to output

translate_coords  Translates a point with coordinates x, y from origin x_offset, y_offset to origin x_origin, y_origin, with an expansion factor (factor == 1 will give a direct mapping)
Appendix E: Knowledge Base Parser

PARSER - the yacc parser of the spatial system

FUNCTION

The parser parses an input data file in order to store data in the lisp system. In addition to syntactic error checking, the parser builds lisp objects given syntactically and semantically correct data. These objects are to serve as data for the spatial reasoning system.

The parser of the spatial system is a C function that may be called from lisp. The C function in turn calls yyparse(), a UNIX system-defined function name that invokes a Yacc program, written in the yacc language. The yacc program produces a Lr(1) parser. The yacc program makes use of a lexical analyzer written in lex. The parser requires as input a file of data whose syntax conforms to the grammar described by the SPATIAL 1.0 Grammar by Barry T. Perricone (Software A & E Confidential), dated August 16, 1985. As the parser parses the input data, certain information is stored in the lisp system that will be operated upon by the spatial reasoning system.

INPUT FILES

The parser expects an input file from which it will read data. The name of the input file is a parameter to the parser.

OUTPUT FILES

None.

COMPILATION

A Makefile, Makeparser, will invoke lex (for the lexical analyzer), yacc (for the parser), and perform the special kind of C compilation needed for a function that will call lisp from C (see Lisp_C.doc for an explanation of the C compilation needed). Certain files resident on the VAX are needed for the lisp-c interface. These files currently reside on the ETL VAX-780 in /src/usr/ucb/lisp/franx/h and /src/usr/ucb/lisp/franx/vax. Once compiled, the parser may be loaded into the lisp system by the cfasl command:

(cfasl "../../reloc/spatial.o "call_yyparse "callparser "integer function")

DIRECTORY STRUCTURE

The following directories pertain to the parser:
NEW-SES/PARSER
Top-level directory, containing the Makefile (Makeparser), and some of the auxiliary lisp files used for justification and display of instantiated lisp objects

- **data**
  - data files (if any) for the parser

- **defs**
  - contains y.tab.h, definition file created by yacc

- **doc**
  - documentation, Lisp_C.doc (the lisp-c interface), and franz.doc (description of franz lisp by John Foderaro)

- **interm**
  - contains intermediate files produced by lex (lex.yy.c), yacc (y.tab.c)

- **reloc**
  - contains relocatables produced by lex (lex.yy.o), yacc (y.tab.o), and the c compiler (spatial.o, the parser in final relocatable form)

- **source**
  - source code for yacc (spatial.y, and its archive, spatial.y,v), lex (mlxspatial.c, and its archive, mlxspatial.c,v)

DOCUMENTS

Essential to the understanding of the parser is the description of the grammar in BNF form as described in the SPATIAL 1.0 Grammar by Barry T. Perricone, dated August 16, 1985 (Software A & E Confidential). Since the parser builds lisp objects, an understanding of the lisp-c interface is essential, and is described in the Lisp_C.doc document written by J. K. Potts (Software A & E).

FUNCTIONS

The following is a list of internal functions in the yacc program with brief descriptions:

- **add_feature**
  - adds feature to the current instantiation of an object

- **add_prim_list**
  - adds gensym to *usr_prim_lists*

- **add_value**
  - adds value to the current instantiation of an object

- **call_yyparse**
  - the c invoked by lisp function that calls the parser

- **clean_up_var**
  - clean up variables

- **command_parse**
  - parses command

- **complete_model**
  - completes the slots of an instantiation
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>free_nameptr</td>
<td>Frees the storage allocated to a nameptr variable</td>
</tr>
<tr>
<td>get_atom_value</td>
<td>Gets actual Lisp value of an atom, given its pname (pname is a printable string)</td>
</tr>
<tr>
<td>get_feature</td>
<td>Gets feature slot of the named object</td>
</tr>
<tr>
<td>get_operator</td>
<td>Gets Lisp value of string representing operators such as &quot;le&quot;, &quot;gt&quot;, etc.</td>
</tr>
<tr>
<td>get_slot_val</td>
<td>Gets slot value of a given slot for the named object</td>
</tr>
<tr>
<td>hashy</td>
<td>Returns a hash code index for a string</td>
</tr>
<tr>
<td>init_var</td>
<td>Initializes variables</td>
</tr>
<tr>
<td>install</td>
<td>Installs a string, its object definition, discipline and nameptr in the hashtable</td>
</tr>
<tr>
<td>install_slot</td>
<td>Installs slot value of a given slot for the current gensym</td>
</tr>
<tr>
<td>lisp_print</td>
<td>Prints any type of valid Lisp object</td>
</tr>
<tr>
<td>locate_name</td>
<td>Calls Oinfo_manage with 'locate parameter in order to locate a Lisp object indexed by its name Lisp name (a list of atoms representing its name)</td>
</tr>
<tr>
<td>lookup</td>
<td>Looks up a string in the hash table. If not there, returns NULL. Otherwise, returns pointer to hash table data object</td>
</tr>
<tr>
<td>make_gensym</td>
<td>Makes a gensym</td>
</tr>
<tr>
<td>make_lisp_name</td>
<td>Makes a Lisp name (a list of atoms) out of a nameptr variable</td>
</tr>
<tr>
<td>make_name_sym</td>
<td>Calls Mkname_sym with an indexing letter (e.g., &quot;U&quot;, &quot;S&quot;)</td>
</tr>
<tr>
<td>make_sym</td>
<td>Calls Mkname_sym with no indexing letter (i.e., &quot;U&quot;, &quot;S&quot;)</td>
</tr>
<tr>
<td>namecopy</td>
<td>Makes a nameptr copy of a nameptr variable</td>
</tr>
<tr>
<td>newstrcat</td>
<td>Concatenates two strings, inserting a space between them</td>
</tr>
<tr>
<td>prim_parent</td>
<td>Finds parent of primitive (or application object) and stores in parent slot of current gensym</td>
</tr>
<tr>
<td>print_hash_table</td>
<td>Prints information about constants, primitives, application objects, and initial models that have been parsed</td>
</tr>
</tbody>
</table>
retrieve_slot
set_feature
set_sym
store_constant
strsave
yyerror

retrieves a slot for the named object
sets feature
sets the input gsym to the given value
stores constant, its type and value in the lisp system. It calls Osksys and Dinfo_manage
returns a fresh copy of string
yyerror prints out errors encountered during parsing
ADDENDA

SPATIAL 1.0 GRAMMAR (BNF FORM)
Spatial 1.0 Grammar (BNF Form)

B.T. Perricone

Software A&E, Inc.
### Table of Contents

*Introduction*
- Notation Conventions ........................................ 3
- Comment Syntax for Spatial 1.0 Knowledge Base ... 3

Main Spatial Knowledge Base Sections .................. 4
- Constants Section (CNST) ...................................... 4
- Spatial Schemas (SCHEMA) .................................. 4 - 7

Initial Model Declaration (MODEL) ........................... 8
- Command Declarations (CMDS) ............................... 9
- Names (NM) .................................................... 10
- Numerics (NUM) ................................................ 11
- Naming Semantics (SEM) ..................................... 12
Introduction

Notation Conventions

[N] Non-terminal symbol
(N) optional single occurrence of N
'N' Literal symbol
"N" String constant
N+ 1 or more occurrences of N
N* 0 or more occurrences of N
N[x,y] minimum occurrences of N is x; maximum is y
I separates alternative syntactical structures. This BNF connector is weaker than a sequence of non-terminal and/or terminal symbols.
N M exclude from the expansion of the non-terminal N the expansions that are possible through the non-terminal M
E represents the empty termination of a non-terminal
(N) denotes the grouping of the syntactical element for a logical syntactical structure.

***N*** comment

There can be zero or more occurrences of "separators" between the syntactical structures of the grammar. "Separators" are not explicitly accounted for within the grammar unless it is important to account for them, in which case it will be noted within the grammar. Multiple occurrences of a "separator" are considered to be a single occurrence of the given "separator". Characters considered to be in the class of "separators" are blank space, tab, newline, and carriage return.

Comment Syntax for Spatial 1.0 Knowledge Base

The comment character for the Spatial 1.0 system described in this document is the backslash character (i.e., \\). Any text on the same line following the backslash character is ignored by the system.
**MAIN SPATIAL KNOWLEDGE BASE SECTIONS (MAIN)**

```plaintext
[spatial_kb] ::= ([constants]) [spatial_schemas] ([initial_model])
[commands]

[constants] ::= 'constants' :: [CNST constant_dcl] (* CNST constant_dcl) *%

[spatial_schemas] ::= (user 'primitives' ::
  [SCHEMA prim_dcl] (* [SCHEMA prim_dcl]) * %)
  'application' 'objects' ::
  [SCHEMA obj_dcl] (* [SCHEMA obj_dcl]) * %

[initial_model] ::= 'initial' 'model' ::
  [MODEL model_dcl] (* [MODEL model_dcl]) * %

[commands] ::= 'actions' :: [CMDS cmd_dcl] (* [CMDS cmd_dcl]) * %
```

**CONSTANTS SECTION (CNST)**

```plaintext
[constant_dcl] ::= [name] [constant_type]

[constant_type] ::= ('string' | 'integer') |
  ('unsigned_int' | 'signed_int')
```

**SPATIAL SCHEMAS (SCHEMA)**

*primitive object schemas*

```plaintext
[prim_dcl] ::= [NM name] ('`
  ('block') :: [pblock] |
  'point' :: [features] |
  'line' :: [features] |
  [SEM prim_nm] :: [SEM legal_dcl])

[pblock] ::= [features] ([height]) ([width]) ([length]) |
  [height] ([width]) ([length]) |
  [width] ([length]) |
  [length]
```

*application object schemas*

```plaintext
[obj_dcl] ::= [NM name] [obj_type]

[no_setof] ::= 'C' ([setof] | [not_setof])
```

*MAIN SPATIAL KNOWLEDGE BASE SECTIONS (MAIN)*

[spatial_kb] ::= ([constants]) [spatial_schemas] ([initial_model])
[commands]

[constants] ::= 'constants' :: [CNST constant_dcl] (* [CNST constant_dcl]) *%

[spatial_schemas] ::= (user 'primitives' ::
  [SCHEMA prim_dcl] (* [SCHEMA prim_dcl]) * %)
  'application' 'objects' ::
  [SCHEMA obj_dcl] (* [SCHEMA obj_dcl]) * %

[initial_model] ::= 'initial' 'model' ::
  [MODEL model_dcl] (* [MODEL model_dcl]) * %

[commands] ::= 'actions' :: [CMDS cmd_dcl] (* [CMDS cmd_dcl]) * %
```

**CONSTANTS SECTION (CNST)**

[constant_dcl] ::= [name] [constant_type]

[constant_type] ::= ('string' | 'integer') |
  ('unsigned_int' | 'signed_int')

**SPATIAL SCHEMAS (SCHEMA)**

*primitive object schemas*

```plaintext
[prim_dcl] ::= [NM name] ('`
  ('block') :: [pblock] |
  'point' :: [features] |
  'line' :: [features] |
  [SEM prim_nm] :: [SEM legal_dcl])

[pblock] ::= [features] ([height]) ([width]) ([length]) |
  [height] ([width]) ([length]) |
  [width] ([length]) |
  [length]
```

*application object schemas*

```plaintext
[obj_dcl] ::= [NM name] [obj_type]

[no_setof] ::= 'C' ([setof] | [not_setof])
```
[not_setof] ::= ('block' ) | [block_dcl] |
      'point' | [point_dcl] |
      'line' | [line_dcl] |
      'conceptual' | [conceptual_dcl] |
      [SEM prim_nm] | [SEM legal_dcl] |
      [SEM obj_nm] | [SEM legal_dcl] |

(setof) ::= 'setof' |
      ('block' ) ( : [block_dcl] ) |
      'point' | [point_dcl] |
      'line' | [line_dcl] |
      'conceptual' | [conceptual_dcl] |
      [SEM prim_nm] | [SEM legal_dcl] |
      [SEM obj_nm] | [SEM legal_dcl] |

[block_dcl] ::= [features] [(height)] [(width)] [(length)] [(origin)] |
      [height] [(width)] [(length)] [(origin)] |
      [width] [(length)] [(origin)] |
      [length] [(origin)] |
      [origin] |

[origin] ::= 'origin' | 'stop' | 'class' | [obj_nm] ' |
      ('constraints' ::
      ( [SEM num_feature_nm] [num_ref] [NUM int_ref] |
        [SEM str_feature_nm] [eq_ref] [NUM str_ref] ) ) ) |

***************************************************************************************
** NOTE: Only one constraint allowed here. For more general case later must be one or more.
***************************************************************************************

[height] ::= [height'] | [NUM pos_int_ref] |
[width] ::= [width'] | [NUM pos_int_ref] |
[length] ::= [length'] | [NUM pos_int_ref] |

[features] ::= ['features' ( [NM name] [feature_type] ) ] |

[feature_type] ::= ('string') | [NM str_ref] |
      'integer' | [NUM int_ref] |

[point_dcl] ::= [features] [origin] | [features] [origin] |
[line_dcl] ::= [features] [origin] | [features] [origin] |

conceptual schema type

[conceptual_dcl] ::= [consists_of] [topology] [relative_origin] |
      ( [structural_constraints] ( [orientation] ) ) |

[consists_of] ::= ['consists_of' :: [consists_nm] ( ['consists_nm'])* ] |
[consists_nm] ::= [name] ( [prim_nm] | 'block' | 'point' | 'line' ) |

Spatial BNF
Version 1.7
January 23, 1986
Page 5
[topology] ::= ['topology' '(' top_dcl ')' (',' top_dcl)* ')
[top_dcl] ::= (NM cof_nm)1 : (NM cof_nm)2 (binary_tolerance)
                   ([binary_constraints])

                             ***********************************************
                             *** (NM cof_nm)1 = (NM cof_nm)2
                             *** (NM cof_nm)1. (NM cof_nm)2 must be declared within the
                             *** consists_of expansion [topology] is associated with
                             ***********************************************

[relative_origin] ::= ['relative' 'origin' 'of' (NM cof_nm)]
[binary_tolerance] ::= ['tolerance' ('x' [min_max])
                      ('y' [min_max])
                      ('z' [min_max])]'

                             ***********************************************
                             *** tolerance between difference in (x y z) of (NM cof_nm)1
                             *** and (x y z) of (NM cof_nm)2
                             ***********************************************
[min_max] ::= ('([NUM int_ref] | [NUM int_ref] | [NUM int_ref])
  
  ******************************************************
  *** [int_ref]₁ ≫ [int_ref]₂  
  ******************************************************

[binary_constraints] ::= ['constraints' | [ln_projection]]

[ln_projection] ::= 'line' 'projection' = ('true' | 'false')

[orientation] ::= ['orientation' = 'parallel' to [SEM line_nm]]

[structural_constraints] ::= ['conceptual' 'constraints']
  
  ([elevation] | [distance])

[elevation] ::= ['maximum' 'elevation' 'difference']
  
  [NM cof nm] ('[NM cof nm][Z,N][NUM pos int_ref]

  ******************************************************
  *** where N is the number of identifiers (i.e., [consists of]
  *** expansions) generated through the [consists of]
  *** expansion ass/w [elevation]
  ******************************************************

[distance] ::= ['distance' ('(x' ('y' ('z') | 'y' ('z') | 'z')
  
  [NM cof nm] ('[NM cof nm][Z,N][NUM pos int_ref]

  ******************************************************
  *** where N is the number of identifiers (i.e., [consists of]
  *** expansions) generated through the [consists of]
  *** expansion ass/w [distance]
  ******************************************************

coordinates

[endpoint] ::= ['endpoint' ['coordinate']

[coordinate] ::= 'x' ::= [NUM pos int_ref]
  
  'y' ::= [NUM pos int_ref]
  
  'z' ::= [NUM pos int_ref]

relations

[num_rel] ::= 'le' | 'ge' | 'lt' | 'gt' | [eq_rel]

[eq_rel] ::= '=' | '!='
**INITIAL MODEL DECLARATION (MODEL)**

```plaintext
[model_dcl] ::= ( ( [SEM ablock_nm] | [SEM aline_nm] | [SEM aconceptual_nm] ) [f_assign] [o_assign] | [SEM apoint_nm] [f_assign] [o_assign] [SCHEMA endpoint] )

[f_assign] ::= ( 'features' [feature_stmt] ( '; [feature_stmt] )'* )

[feature_stmt] ::= ( [SEM feature_nm] = [SEM legal_val] )

[o_assign] ::= ( 'origin' = [SCHEMA coordinate] )
```
• COMMANDS (CMDS)

[cmd_dcl] ::= 'lastmark' | 'mark' | 'rdcommand' [NM file_nm] | 'obtain' [SEM obj_nm] ([obtain_opts]) | 'justify' [j_opts]

• [j_opts] ::= [obj_nm] ( '=' '(' [NUM unsigned_int] [NUM unsigned_int] [NUM unsigned_int] ')' )

*******************************
*** where:
***
*** [NUM unsigned_int]1 --- x origin coordinate
*** [NUM unsigned_int]2 --- y origin coordinate
*** [NUM unsigned_int]3 --- z origin coordinate
*******************************

[obtain_opts] ::= 'within' [boundary]
NAMES (NM)

[str_ref] ::= [string] | (SEM strc_nm)
[file_nm] ::= [letter] [file_end]
[name] ::= [word] (' [separator] [word] ')

***********************************
*** all [name]s are unique
***********************************

[string] ::= a sequence of characters bracketed by double quote character (i.e., "") that does not extend over a newline or carriage return

[word] ::= [letter] ( [word_end] )
[word_end] ::= [digit] | [letter] | . ( [letter] | [digit] )
[file_end] ::= [word_end] | : [word_end]
[letter] ::= A-Z | a-z
[digit] ::= 0-9
[separator] ::= blank space | tab | newline | carriage return
NUMERICS (NUM)

*******************************************************************************

*** all integers generated (e.g. [unsigned_int], [signed_int], [pos_int_ref], etc.)
*** must be in the range specified by [default-int]
*******************************************************************************

[default_int] ::= the range of integers that are supported by the host computer
[unsigned_int] ::= [NM digit]*
[signed_int] ::= ('-') [NM digit]*
[int_ref] ::= [unsigned_int] | [signed_int] | [intc_nmc]
[pos_int_ref] ::= [unsigned_int] | [intc_nmc]

*******************************************************************************

*** if expanded to [intc_nmc] then [intc_nmc] must reference
*** a POSITIVE integer
*******************************************************************************

[zero] ::= the [digit] 0
NAMING SEMANTICS (SEM)

[strc_nm] ::= a unique [NM name] used within a [CNST constant_dcl] expansion to a string type constant

[intc_nm] ::= a unique [NM name] used within a [CNST constant_dcl] expansion to an integer type constant

[obj_nm] ::= a unique [NM name] that was used in an [SCHEMA obj_dcl] expansion

[prim_nm] ::= a unique [NM name] that was used in an [SCHEMA prim_dcl] expansion

[cof_nm] ::= a [NM name] expanded from within a [SCHEMA consists_nm] expansion

[line_nm] ::= a [NM name] that references a LINE-typed object or a descendent of a LINE-typed object.

[ablock_nm] ::= a [NM name] that references a BLOCK-typed [obj_nm] or a descendent of a BLOCK-typed [obj_nm].

[aline_nm] ::= a [NM name] that references a LINE-typed [obj_nm] or a descendent of a LINE-typed [obj_nm].

[aconceptual_nm] ::= a [NM name] that references a CONCEPTUAL-typed [obj_nm] or a descendent of a CONCEPTUAL-typed [obj_nm].

[apoint_nm] ::= a [NM name] that references a POINT-typed [obj_nm] or a descendent of a POINT-typed [obj_nm].

[num_feature_nm] ::= a unique [NM name] used within a [SCHEMA features_dcl] expansion to declare a feature of [SCHEMA features_type] 'integer'

[str_feature_nm] ::= a unique [NM name] used within a [SCHEMA features_dcl] expansion to declare a feature of [SCHEMA features_type] 'string'

[legal_dcl] ::= a legal declaration expansion for the primitive type of the [name] it is associated with (association may be through an 'ancestor' of the [name]).