## Appendix A

### Definitions of a GIS

<table>
<thead>
<tr>
<th>Year of Reference</th>
<th>Author</th>
<th>Definition</th>
</tr>
</thead>
</table>
| 1989              | Arnoff | "any manual or computer-based set of procedures used to store and manipulate geographically referenced data."
| 1989              | Carter | "an institutional entity, reflecting an organisational structure that integrates technology with a database, expertise, and continuing financial support over time."
| 1989              | Koshkariov et al | "a system with advanced geo-modelling capabilities."
| 1989              | Parker | "an information technology which stores, analyses, and displays both spatial and non-spatial data."
| 1988              | Cowen | "a decision support system involving the integration of spatially referenced data in a problem-solving environment."
| 1987              | Smith et al. | "a database system in which most of the data are spatially indexed, and upon which a set of procedures operated in order to answer queries about spatial entities in the databases."
| 1987              | US Department of Environment | "a system for capturing, storing, checking, manipulating, analysing, and displaying data which are spatially referenced to the Earth."
| 1986              | Burrough | "a powerful set of tools for collecting, storing, retrieving, at will, transforming and displaying spatial data from the real world."
| 1986              | Devine & Field | "a form of MIS [Management Information System] that allows map display of the general information."
| 1981              | Ozemoy et al | "an automated set of functions that provides professionals with advanced capabilities for the storage, retrieval, manipulation, and display of geographically located data."
| 1979              | Dueker | "a special case of information systems where the database consists of observations on spatially distributed features, activities, or events, which are definable in space as points, lines or areas. A GIS manipulates data about these points, lines and areas to retrieve data for ad hoc queries and analyses."

Source: Cited in [Mennecke 1997]
Appendix B

The VAX OPS5 Structure

VAX OPS5, a production system language, consists of a declaration section that describes the data objects of the programme, followed by a production section that contains the rules. During execution, the data operated on by the programme are kept in the working memory. The rules are maintained in the production memory. Working memory is usually initialized after the declarations and rules have been loaded. The declaration section contains the definitions of the data object types and all user-defined functions that can be referenced in the rules.

The inference engine in VAX OPS5, consisting of the three algorithms for "match-rules," "select-rules," and "execute-rules," directly supports forward chaining, even though backward chaining can be implemented [Digital 1988].

B. 1. DATA TYPES

VAX OPS5 uses two primitive data types, also known as scalar types: numbers and symbolic atoms. Symbolic atom is any sequence of characters that is not a number and which can be treated as a single unit. The compound data structure type definable in a VAX OPS5 programme is element class. The components of an element class are called attributes. An element class declaration is similar to a structure declaration in Cobol or a record declaration in Pascal.
VAX OPS5 is characterized by:

- A global database.
- Condition-action (or if-then) rules programmed in the form of productions, which operate on the global database.
- Productions that are executed in an unspecified order.
- Computations with symbolic expressions and numbers.
- Simple syntax.

As mentioned earlier (Appendix B, Para 1), VAX OPS5 has two key components: **working memory (wm)** and **production memory**. Working memory is a global database of information pertaining to the problem domain. This information is stored in elements which are grouped into classes. Elements storing similar information can be grouped in the same class. WM can contain several classes of elements, and each class can have more than one element. WM is dynamic. While an OPS5 programme is executed, elements are added, deleted, or modified, continually.

A working-memory element (wme) is a sequence of atoms that represents an object or a concept. Each atom is stored in a field that can be labelled with an attribute name. A wme can be represented using a combination of the following:

- A class-name;
- A list of scalar attributes and their values;
- A vector attribute and its value.

The class-name identifies an element's class, and the attributes and their values describe the element's characteristics. The value of each scalar attribute is an atom. The value of a vector attribute can be one or more atoms. The format for specifying a wme is as follows:

```
[class-name] [(scalar-attribute value)...] [vector-attribute value]
```
A class-name is a symbol that identifies a group of similar elements. Elements that have the same class-name have the same attributes, but the values of the attributes are different.

An attribute consists of the attribute designator (^), followed by the name of the attribute which describes the element’s characteristics.

The value of a scalar attribute is an atom, e.g., ^number 102

The value of a vector attribute is a list of one or more atoms, e.g., ^date 2 Jan 1988

**B. 2. INTERNAL REPRESENTATION OF WORKING-MEMORY ELEMENTS**

Working memory elements are marked with a time tag or recency attribute, which is used as a part of conflict resolution. Larger values of the time tag indicate more recent elements. When an element in working-memory gets modified the element gets a new time tag. When the OPS5 "examine working memory" command (wm) is issued, each working memory element is displayed preceded by an integer, which is the time tag for the element.

The first field of the structure that holds a wme is reserved for the class-name of the element. If an element does not have a class-name, the run-time places NIL in the first field. The compiler assigns fields to the names of the scalar attributes when the attribute names are declared. An attribute’s field stores that attribute’s value. The field assigned to each attribute name is global meaning that, the attribute name refers to the same field for each element-class in which the attribute name appears.
Declarations: Attribute names and external routines must be declared using the literalize, literal, vector-attribute, or external declaration, and they must appear before other kinds of statements of the programme. Literalize declaration does the following:

associates a class with a list of attribute names;
signals to the compiler to assign unique fields to the specified attribute names

(literalize road
  name
  start-point
  end-point
  junctions)

The above declaration associates the class-name “road” with the attribute names, “name,” “start-point,” “end-point,” and “junctions.” This declaration is also an indication to the compiler to assign appropriate fields to the attribute names.

B. 3. PRODUCTIONS

Each production is of the form

(p production-name
  (condition-element-1)
  (condition-element-2)
  ...
  (condition-element-n)
  -->
  (action-1)
  (action-2)
  ...
  (action-m))

The p that follows the opening parenthesis signifies that the code following it is a production. The “production-name” distinguishes the production from other productions in a programme. The LHS of the production is separated from the RHS
by an arrow The name of each production within a programme must be unique. A production can be named with any string except “NIL.”

B. 3. 1. The Left-Hand Side

Condition-elements on the LHS of the production can be positive or negative. The first condition-element must be positive. A LHS can have a maximum of 64 positive condition-elements and any number of negative condition-elements. The values that can be specified in the components of a condition-element can be constant atoms, variables or function calls. They can be preceded by a predicate indicating the comparison operation (equal to, greater than, *et cetera*) to be performed.

In OPS5, constants can be symbols, integers, or real numbers. A variable is a symbol enclosed in angle brackets (< >).

*Conjunctions:* A conjunction is a pattern of conditional tests (similar to logical AND), all of which must be true of an atom in a working memory element. A conjunction can be specified by enclosing the list of conditional tests in braces ( { } ).

*Disjunctions:* A disjunction is a pattern containing a list of constant atoms. Only the atoms specified in the list can match the pattern. A disjunction is similar to logical inclusive OR. A disjunction can be specified by enclosing the list of constant atoms between double angle brackets (<< >>).

B. 3. 2. The Right-Hand Side

The RHS of a production consists of one or more actions. Actions perform the following operations:
Modify working-memory
Save and restore the state of working-memory and the
conflict set.
Stop programme execution
Bind variables
Manipulate files
Write output
Control loops
Add productions to executing programmes.
Call external sub-programmes

An action includes an action-name and its attributes enclosed within
parentheses.

B. 4. CONFLICT RESOLUTION STRATEGIES

Instantiations are the objects in the conflict set. An instantiation is an ordered
pair consisting of a production name and a list of working-memory elements satisfying
the production's LHS. A conflict set is a collection of ordered pairs of the form:

< Production-name, List of elements matched by production's LHS >.

During the conflict resolution step, the OPS5 interpreter examines the conflict
set to find an instantiation that dominates all the others under a set of ordering rules
called a "conflict resolution strategy" OPS5 provides two such strategies, LEX and
MEA [Brownston et al. 1985].

When more than one rule is created in the conflict set, conflict resolution is
achieved using one of the two strategies: LEX and MEA. LEX is the simpler of the
two. Both the strategies use the following concepts:
Appendix B. The VAX OPSS Structure 176

**Refraction.** Selects an instantiation only once. Refraction prevents programmes from looping indefinitely on the same data by removing instantiations from the conflict set after they have been selected.

**Recency.** Selects the instantiations that refers to the most recent data in working-memory, as indicated by the time tag attached to each wme (vide Section B.2). Therefore, the system must select the instantiation that contains the highest time tag.

**Specificity.** Selects an instantiation of a production whose LHS is the most specific. Specificity of each production in a programme is calculated by counting the number of condition tests on the LHS. The production with the maximum number of conditional tests is the most specific.

Both strategies, LEX and MEA, apply the rules in the following order: refraction, recency, specificity. However, MEA strategy includes an extra step after refraction, which helps to organize large programmes. This step orders the instantiation according to the recency of the wme, matching the first condition-element in each production. The default strategy for VAX OPS5 is LEX. This can be changed to MEA, if required, through the **strategy** command with MEA as the keyword.

**LEX Strategy.** LEX is suitable for programmes that do not depend on the order in which the productions are executed—the normal case for production systems. LEX uses the following rules in sequence to order the instantiations in the conflict set.

Apply refraction by removing from the conflict set the instantiations the run-time system has selected during the previous cycle.
Order the rest of the instantiations according to their recency, and select the instantiation with the highest level of recency. If more than one instantiation has the highest level of recency, order those instantiations according to their specificity, and select the instantiation with the highest level of specificity. If more than one instantiation has the highest level of specificity, select an instantiation arbitrarily.

**MEA Strategy.** MEA places the highest priority on the production whose first condition-element is matched by the most recent wme. This is most appropriate when the most important condition-element is placed first on the LHS of each production. MEA uses the following rules in sequence to order the instantiations in the conflict set.

1. Apply refraction by removing from the conflict set the instantiations the run-time system has selected during the previous cycle. Compare the first time tag of each instantiation still remaining in the conflict set and select the instantiation with the highest level of recency. If more than one instantiation has the highest level of recency for the first time tag, order those instantiations according to their recency (using all time tags) and select the instantiation with the highest level of recency.

After the run-time system has selected an instantiation, the recognize-act cycle enters the “execute” phase. During this phase, variables are bound to values, and the actions on the RHS of the productions to which the selected instantiations refer are executed.

**B. 5. ORGANIZATION AND CONTROL IN OPS5**

By exploiting the built-in capabilities of OPS5, a production system designer can exert the desired degree of control over the execution of the system. In OPS5, the built-in structured data types are element-classes and vector attributes. The `substr` function can be used to implement data structures such as arrays, strings, stacks, queues, and deques, depending on the set of operations permitted on the vector
Because production systems access data by value, rather than by address, it is easier to create linked data structures in OPS5. Links are traced by the process of matching rather than dereferencing, and this makes the linked data structures easier to manage. Linking allows implementation of graphs such as goal trees, semantic networks, frames and other complex structures.

The OPS5 interpreter does not simply test the expressions to determine whether they are true; rather it engages in a search to determine if it can bind variables in such a way as to make the expression true. It is this characteristic of pattern matching, along with dynamic size of working-memory that gives production systems their distinctive capabilities.

**B. 6. LEARNING IN OPS5**

Learning is any change in behaviour of a system that contributes to the enhancement of the system’s performance. There are different approaches to machine learning. Early programmes affected learning through self modification of stored parameters. In the production system model, incremental changes in performance are effected by adding new production rules to an existing rule base. Researchers have also investigated a form of learning that results from changes in working-memory and reorganisation of existing rules [Brownston et al. 1985].

Learning can be accomplished in two ways, either by remembering the results of previous search or by learning the definition of an object more precisely so that the search space may be pruned rapidly. Learning, therefore, may be classified as either rôte learning or inductive learning.
B. 6. 1. Rôte Learning

Rôte learning allows the system to memorise the examples of an object for which it has already searched so that, when it is asked to search for the same object again, it retrieves the previous examples instead of searching again. It stores only pre-defined high level objects. Known examples are stored in a database that is separate from the spatial object KB. The decision on whether an example should be stored or not depends on factors such as complexity of the object and the frequency with which they are sought.

B. 6. 2. Inductive Learning

Inductive learning is used to provide a new definition for an object from a given set of examples so that the search for an object can proceed more efficiently. To learn a new definition of an object, the user can give example definitions, by specifying the appropriate values, and the system generates the definitions using these properties.

B. 6. 3. Strengthening and Weakening

One way of implementing learning in a production system is to tune a set of parameters so that they converge on a set of optimal values. A number can be associated with each rule to indicate the desirability of its firing. The probability that this particular rule will fire can be raised by increasing this number. Such a number is often called the rule’s strength. Naturally enough, incrementing the strength is called \textit{strengthening} and decrementing the strength is called \textit{weakening}. 
B. 6. 4. Generalization

A system that learns by generalization, starts with a set of overly specific rules and generates new rules that are more general and abstract. These inferred rules may improve performance even if they are incorrect, although they are not guaranteed to do so. If the system overgeneralizes, it may be necessary to backtrack or to specialize some of the rules that are too general.

B. 7. SUMMARY

Many people associate knowledge-based systems with rule-based systems. We can have knowledge-intensive problem solving without using rule-based representation and inferencing, although rules or implications are indeed one of the most common ways of expressing knowledge. Rule-based systems are a kind of production system that utilizes rules of deductive reasoning to reach logical conclusions. Production systems are used most frequently by interpreting these rules (1) as condition-action rules in forward-chaining control or (2) as sets of logical implications, from which deductions are drawn, in backward-chaining control.

OPS5 is a forward-chaining production system development language. The use of sophisticated indexing scheme, known as the Rete algorithm, for finding rules that match the current rule-base makes OPS5 one of the tools that executes the fastest.

When more than one rule become qualified to be fired, two OPS5 conflict-resolution strategies are available: LEX and MEA. LEX is the simpler of the two. MEA (means end analysis) places additional emphasis on the recency of a wme that matches the first rule condition.
Appendix C

C1. The START Menu

SUBROUTINE startmenu()
C Subroutine for displaying the START Menu
implicit integer (a-z)
include 'opslibrary:opsdef.for'
include 'syslibrary:uisentry'
include 'syslibrary:uisusrdef'
integer*4 colind, keybuf
character(*) title
common /cbl/vd_idl, wd_idl, vd_id2, wd_id2, vd_id5, wd_id5, keybuf
common /cd/vd_idm, wd_idm
external create_window1, kerala1, hilights1, hilight2, delwidm
parameter (title = 'Start Menu')

call create_window1(vd_idm, wd_idm, -1.0, -1.0, 4.0, 0.0, 2.0, 5.0, 3.0, title, -1.0, -1.0, 4.0, 0.0, 2.0, 5.0, 3.0, 0.0, 0.0)

call uis$set_color(vd_idm, 0.0, 0.8, 0.9, 0.0)
call uis$set_color(vd_idm, 1.0, 1.0, 0.0, 0.0)
call uis$set_writing_index(vd_idm, 0.9, 1.0)
call uis$text(vd_idm, 0.0, '1.Kerala', 0.0, 1.5)
call uis$text(vd_idm, 0.0, '2.Exit', 0.0, 0.5)

call uis$set_pointer_ast(vd_idm, wd_idm, hilights1, -0.1, 0.1, 1.0, 0.8, 0.9, 0.0)
call uis$set_button_ast(vd_idm, wd_idm, kerala1, keybuf, -0.1, 0.1, 0.4, 0.1, 5.0)
call uis$set_pointer_ast(vd_idm, wd_idm, hilights1, -0.1, 0.0, 4.0, 0.0, 5.0)
2 hilight2,%ref(%loc(vd_idm)))
call uis$set_button_ast(vd_idm, wd_idm, kerala1, keybuf, -1.0, 0.0, 0.4, 0.0, 5.0)
call uis$set_button_ast(vd_idm, wd_idm, delwidm, keybuf, -1.0, 0.0, 0.4, 0.0, 5.0)
return
end
C2. Selecting the Trivandrum District

SUBROUTINE kerala()

C Displays the map of Kerala with all districts from which the Trivandrum District is selected

implicit integer (a-z)
include 'sys$library:uisentry'
include 'sys$library:uisusrdef'
common /cb1/vd_id1,wd_id1,vd_id2,wd_id2,vd_id5,wd_id5,keybuf
common /cb4/keycode,vd_id6,wd_id6,vd_ide,wd_ide
common /cb/vd_idm,wd_idm
character*20 rdname,rdname
real x1,y1,x2,y2
character(*) title
external plotlinek,plotpointk,create_windowl,hmenu,tvm
parameter (title = 'KERALA STATE')

call create_windowl(vd_id6,wd_id6,0.0,0.060.0,60.0,25.0,25.0,8.0,15.0)
call hcnis$read_display(vd_id6,'kerala.uis')
rd_name = 'dstnames.dat'
call plotpointk(rd_name,vd_id6)
call uis$set_text(vd_id6,0,'Move the pointer inside the district boundary ',0.0,60.0)
call uis$set_text(vd_id6,0,'Click "left mouse button" to select',1.0,60.0)
call uis$set_text(vd_id6,0,'Click "Right mouse button" to Delete Map'.1.0,58.0)
call uis$set_button_ast(vd_id6,wd_id6,twn,keybuf,41.48,0.78,47.85,6.36)
return
end

SUBROUTINE twn()

C Invoking the Trivandrum District Menu

implicit integer (a-z)
include 'sys$library:uisentry'
include 'sys$library:uisusrdef'
common /cb1/vd_id1,wd_id1,vd_id2,wd_id2,vd_id5,wd_id5,keybuf
common /cb4/keycode,vd_id6,wd_id6,vd_ide,wd_ide
common /cb/vd_id7,wd_id7
external kerala

if (keybuf .eq. uis$c_pointer_button_3) then
  call uis$delete_window(wd_id6)
call startmenu
else if (keybuf .eq. uis$c_pointer_button_1) then
  call uis$delete_window(wd_id6)
call trivandum
endif
return
end

SUBROUTINE trivandum()

C Displays the map of Trivandrum District

implicit integer (a-z)
include 'sys$library:uisentry'
include 'sys$library:uisusrdef'
integer(*) title
common /cbl/vd_id1,wd_id1, vd_id2, wd_id2, vd_id5, wd_id5, keybuf
common /cb4/keycode, vd_id6, wd_id6, vd_idc, wd_idc
common /cbc/vd_id7, wd_id7
external plotline, plotpoint, create_window, tvmcity
data vcm_size/16/
parameter (title = 'Trivandrum District')
lwidth = 1.0
call create_window1(vd_id7, wd_id7, 0.0, 0.0, 60.0, 60.0, 25.0, 25.0, title,
    40.0, 0.0, 50.0, 10.0, 25.0, 25.0, 8.0, 3.0)
2
rname = 'tvm.dat'
call plotline (rname, vd_id7)
call uis$set_color (vd_id7, 0, 0.8, 0.8, 0.0)
call uis$set_color (vd_id7, 1, 1.0, 0.0, 0.0)
call uis$set_writing_index (vd_id7, 0, 8, 1)
call uis$set_line_width (vd_id7, 8, 9, lwidth, wdpl$C_width_world)
call uis$set_text(vd_id7, 8, 'TRIVANDRUM DISTRICT', 45.0, 4.8)
call uis$set_text(vd_id7, 0, 'Click "left mouse button" to select', 40.0, 10.50)
call uis$set_text(vd_id7, 0, 'Click "Right mouse button" to Delete Map', 40.0, 10.0)
call uis$set_button Ast(vd_id7, vd_id7, tvmcity, keybuf, 40.0, 0.0, 50.0, 10.0)
return
end
C3. Code for Trivandrum City Menu

SUBROUTINE tvmcity()
C Invokes the Trivandrum city menu
implicit integer (a-z)
include 'sys$library:uisentry'
include 'sys$library:uisusrdef'
integer(*) title
common /cb/vd_idt,wd_idt
common /cb1/vd_id1,wd_id1,vd_id2,wd_id2,vd_id5,wd_id5,keybuf
common /cbc/vd_id7,wd_id7
external tvmmenu, kerala
parameter (title = 'Thiruvananthapuram City')
if (keybuf .eq. uis$c_pointer_button_3) then
  call uis$delete_window(wd_id7)
call kerala
else if (keybuf .eq. uis$c_pointer_button_1) then
  call uis$delete_window(wd_id7)
call tvmmenu
end if
return
end

SUBROUTINE tvmmenu()
C Displays menu for Trivandrum City
implicit integer (a-z)
include 'sys$library:uisentry'
include 'sys$library:uisusrdef'
character(*) title
integer ovls\w1'tch
common /kb/kb_id,vcm_id1
common /cb/vd_idt,wd_idt
common /cb1/vd_id1,wd_id1,vd_id2,wd_id2,vd_id5,wd_id5,keybuf
common /cbc/vd_id7,wd_id7
common /os/ ovls\w1'tch
external test,test1,test3,hi\lghtc.hilight2, util
data vcm_size/16/
parameter (title = 'Thiruvananthapuram City')
ovls\w1'tch = 0
call create_windowl (vd_idt,wd_idt,0.0,0,0,8.0,3.0,8.0,3.0,title,
  0.0,0,8.0,3.0,8.0,3.0,0,0,0,0,0)
call uis$set_color (vd_idt,0.1,0.1,0.0,0)
call uis$set_color (vd_idt,1.1,0.0,0,0)
call uis$set_writing_index (vd_idt,0,9,1)
call uis$text(vd_idt,0,0,1.1,0.2,5) call uis$text(vd_idt,0,1.2,5)
call uis$text(vd_idt,0,3. Exit this Menu ',0.0,1.0)
call uis$set_pointer_ast(vd_idt,wd_idt,hl\lghtc., 0.0,2.0,8.0,2.5,hl\lught2,%ref(%loc(vd_idt)))
call uis$set_button_ast(vd_idt,wd_idt, test1., keybuf,0.0,2.0,8.0,2.5)
call uiss$set_pointer_ast(vd_idt,wd_idt, hilightc,, 0.0,1.0,0.8,0.1,5,hilight2,%ref(%loc(vd_idt)))
call uiss$set_button_ast(vd_idt,wd_idt, test3,, keybuf,0.0,1.0,0.8,0.1,5)
call uiss$set_pointer_ast(vd_idt,wd_idt, hilightc,, 7.0,0.0,7.5,0.5,hilight2,%ref(%loc(vd_idt)))
call uiss$set_button_ast(vd_idt,wd_idt, test,, keybuf,7.0,0.0,7.5,0.5)
call util
return
end
C4. Code for Zooming Selected Area of the Map

SUBROUTINE gozoom()

C Zooming a selected area of the map
implicit integer (a-z)
include 'sys$library:uisen8try'
include 'sys$library:uisusdef'
integer*4 keycode,button
integer box_start_flag,switch
character*15 hcopy
real*4 mfact
real*4 xmin,ymin,xmax,ymax,sty,x1,y1,x2,y2
common /cb1/xd_id1,yd_id1,xd_id2,yd_id2,xd_id5,yd_id5,keybuf
common /ch2/button,keyst
common /blk1/box_start_flag
common /blk2/xmin,ymin,xmax,ymax,sty,x1,y1,x2,y2
common /sw/switch
common /mul/mfact
common /zmw/d_id/d_id
common /sws/fswitch,tswitch,pswitch,mswitch
common /hc/hcopy
external zoomsize,create_window2

if (button .eq. uissc_pointer_button_l) then
  if (zswitch .lt. 1) then
    zswitch = 1
    call create_window2(vd_idz,wd_idz,0.0,0.0,12.0,0.0,0.0,12.0,1.0,0.0,0.0,19.0,24.7)
  end if
  call uiss$set_color(vd_idz,0.0,0.0,0.0,0.0)
  call uiss$set_color(vd_idz,1.1,0.1,0.0,0.0)
  call uiss$set_writing_index(vd_idz,0.1,1.1)
  call uiss$plot(vd_idz,1.0,0.0,0.0,1.0,0.0,0.0,1.1,9.0,0.9,0.0,1.0,9)
  call uiss$plot(vd_idz,1.1,5.0,1.1,5.0,9)
  call uiss$plot(vd_idz,1.3,0.0,1.3,0.0,9)
  call uiss$plot(vd_idz,1.4,5.0,1.4,5.0,9)
  call uiss$plot(vd_idz,1.6,0.0,1.6,0.0,9)
  call uiss$plot(vd_idz,1.7,5.0,1.7,5.0,9)
  call uiss$plot(vd_idz,1.9,0.0,1.9,0.0,9)
  call uiss$plot(vd_idz,1.10,5.0,1.10,5.0,9)
  call uiss$plot(vd_idz,1.12,0.0,1.12,0.0,9)
  call uiss$set_color(vd_idz,2.0,1.0,1.0,0.0)
  call uiss$set_writing_index(vd_idz,2.3,2.3,2)
  call uiss$set_char_size(vd_idz,3.4,0.4,0.4)
  call uiss$set(wd_idz,4.0,10x\'0.2,0.8)
  call uiss$set(wd_idz,4.0,20x\'1.7,0.8)
  call uiss$set(wd_idz,4.0,40x\'3.2,0.8)
  call uiss$set(wd_idz,4.0,50x\'4.7,0.8)
  call uiss$set(wd_idz,4.0,75x\'6.2,0.8)
  call uiss$set(wd_idz,4.0,100x\'7.55,0.8)
  call uiss$set(wd_idz,4.0,150x\'9.05,0.8)
SUBROUTINE zoomsizet()
Selects the zooming ratio
implicit integer (a-z)
include 'syslibrary.usenter'
include 'syslibrary.ususrdef'
real*4 mfact,x,y
real*4 xmin,ymin,xmax,ymax,x1,y1,x2,y2
common /cb1/vd_id1,wd_id1,vd_id2,wd_id2,vd_id5,wd_id5,keybuf
common /cb2/button,keylist
common /blk1/box_start_flag
common /blk2/xmin,ymin,xmax,ymax,x1,y1,x2,y2
common /sw/mswitch
common /mul/mfact
common /m/vd_id5,wd_id5
common /sws/s_with,ts_with,pswitch,zswitch,mswitch
external update
if (keybuf .lt. 0) then
  call uis$delete_position(vd_idz,wd_idz,x,y)
call uis$delete_window(vd_idz)
zswitch = 0
box_start_flag = 0
if ((y .gt. 0.0) .and. (y .lt. 1.0)) then
  if ((x .gt. 0.0) .and. (x .lt. 1.5)) then
    mfact = 0.10
  else if ((x .gt. 1.5) .and. (x .lt. 3.0)) then
    mfact = 0.20
  else if ((x .gt. 3.0) .and. (x .lt. 4.5)) then
    mfact = 0.40
  else if ((x .gt. 4.5) .and. (x .lt. 6.0)) then
    mfact = 0.50
  else if ((x .gt. 6.0) .and. (x .lt. 7.5)) then
    mfact = 0.75
  else if ((x .gt. 7.5) .and. (x .lt. 9.0)) then
  else
    mfact = 1.0
  endif
endif
endif
else if ((x.gt.9.0) .and. (x .lt. 10.5)) then
mfact = 1.5
else if ((x.gt.10.5) .and. (x .lt. 12.0)) then
mfact = 2.0
end if

call uis$set_button_ast('d_id1.wd_id1.update..button.xmin.ymin.xmax,ymax)
end if
end if
return
end

SUBROUTINE update()
Allows user to select an area of the map
implicit integer (a-z)
include 'sys$library:uisentry'
include 'sys$library:uisuserdef'
integer box_start_flag
real*4 xmin,ymin,xmax,ymax,x1,y1,x2,y2
common /cb1/vd_id1,wid_id1,wid_id2,wid_id2.wid_id5.wid_id5.keybuf
common /cb2/button.keylist
common /blk1/box_start_flag
common /blk2/xmin,ymin,xmax,ymax,x1,y1,x2,y2
common /mul/mfact
external rubberband,gdum
if (button .eq. uis$c_pointer_button_1) then
call uis$get_pointer_position('d_id1.wd_id1.stx.sty)
call uis$delete_window('d_id2)
x1 = stx
x2 = stx
y1 = sty
y2 = sty

C Draw first "degenerate" box
call uis$plot('d_id1.wd_id1.x1.y1,x2.y2.x1.y2,x1,y1)
call uis$set_button_ast('d_id1.wd_id1..xmin,ymin,xmax,ymax)

C Set up pointer ast for rubberband
call uis$pointer_ast('d_id1.wd_id1.rubberband,xmin,ymin,xmax,ymax.)
call uis$set_button_ast('d_id1.wd_id1.gdum.keybuf,xmin,ymin,xmax,ymax)
end if
return
end

SUBROUTINE gdum()
implicit integer (a-z)
include 'sys$library:uisentry'
include 'sys$library:uisuserdef'
integer box_start_flag
real*4 xmin,ymin,xmax,ymax,x1,y1,x2,y2
common /cb1/vd_id1.wid_id1.wid_id2.wid_id2.wid_id5.wid_id5.keybuf
common /cb2/button.keylist
common /blk1/box_start_flag
common /blk2/xmin,ymin,xmax,ymax,x1,y1,x2,y2
common /mul/mfact
common /hcopy
SUBROUTINE rubberband( )

implicit integer (a-z)
include 'syslibrary.usentry'
include 'syslibrary.ususrdef'
integer box_start_flag.button.keybuf
real*4 x1,y1,x2,y2
common /cb1/vd_id1.wd_id1.vd_id2.wd_id2.vd_id5.wd_id5.keybuf
common /cb2/button.keylist

if (box_start_flag .eq. 1) then
    call uis$set_pointer_ast(vd_id1.wd_id1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.
SUBROUTINE zoomstop1()

  implicit integer (a-z)

  include 'sys$library/uisensry'
  include 'sys$library/uisusrdef'

  common /cb2/button.keycode
  common /cb3/wd_id3,wid_id3,wid_id9,wid_id9,wid_id10,wid_id10
  common /cb5/zx1.zx2.zy1.zy2

  integer*4 code_1
  real*4 abs_pos_x
  integer*4 code_2
  real*4 abs_pos_y
  integer*4 end_of_list

end structure
end record

placelist2.code_1 = wdplSc.abs_pos_x
placelist2.abs_pos_x = 16.0
placelist2.code_2 = wdplSc.abs_pos_y
placelist2.abs_pos_y = 10.0
placelist2.end_of_list = wdplSc.end_of_list

if (keycode .eq. uis$c_pointer_button_1) then

  status = uis$get_pointer_position(vd_id1.wd_id3.x1.y1)
  zx11 = x1 + 2.0
  zy11 = y1 + 2.0
  zx21 = x1 + 2.0
  zy21 = y1 + 2.0

  if (zx11 .lt. 0.0) then
    zx11 = 0.0
  end if

  if (zy11 .lt. 0.0) then
    zy11 = 0.0
  end if

  if (zx21 .gt. zx11) then
    zx21 = zx11
  end if

  if (zy21 .gt. zy11) then
    zy21 = zy11
  end if

  wd_id4 = uis$create_window(vd_id1,'sys$workstation' 'zoomed View'.zx11.zy11.
                       zx21.zy21,10.0,10.0.placelist2)
  call uis$pop_view_port(wd_id3)
  call uis$set_buttonasm(vd_id1.wd_id3.zoomstop1.keycode.zx1.zy1.zx2.zy2)
end if

return
end

SUBROUTINE zoomstop1()

  implicit integer (a-z)

  include 'sys$library/uisensry'
  include 'sys$library/uisusrdef'

  common /cb2/button.keycode
  common /cb3/wd_id3,wid_id3,wid_id9,wid_id9,wid_id10,wid_id10
  common /cb5/zx1.zx2.zy1.zy2

  integer*4 code_1
  real*4 abs_pos_x
  integer*4 code_2
  real*4 abs_pos_y
  integer*4 end_of_list

end structure
end record

placelist2.code_1 = wdplSc.abs_pos_x
placelist2.abs_pos_x = 16.0
placelist2.code_2 = wdplSc.abs_pos_y
placelist2.abs_pos_y = 10.0
placelist2.end_of_list = wdplSc.end_of_list

if (keycode .eq. uis$c_pointer_button_1) then

  status = uis$get_pointer_position(vd_id1.wd_id3.x1.y1)
  zx11 = x1 + 2.0
  zy11 = y1 + 2.0
  zx21 = x1 + 2.0
  zy21 = y1 + 2.0

  if (zx11 .lt. 0.0) then
    zx11 = 0.0
  end if

  if (zy11 .lt. 0.0) then
    zy11 = 0.0
  end if

  if (zx21 .gt. zx11) then
    zx21 = zx11
  end if

  if (zy21 .gt. zy11) then
    zy21 = zy11
  end if

  wd_id4 = uis$create_window(vd_id1,'sys$workstation' 'zoomed View'.zx11.zy11.
                       zx21.zy21,10.0,10.0.placelist2)
  call uis$pop_view_port(wd_id3)
  call uis$set_buttonasm(vd_id1.wd_id3.zoomstop1.keycode.zx1.zy1.zx2.zy2)
end if

return
end
end if
return
end
C5. Code for Important Roads and Rivers Menu

SUBROUTINE rdmenu( )
C Program fragments that displays the distribution of major roads
implicit integer (a-z)
include 'sys$library:uisentry'
include 'sys$library:uisusrdef'
character(*) title
common /cb1/"v_d_id1, v_d_id2,v_d_id2,v_d_id5,v_d_id5,keybuf
common /cb2/button.keycode
common /vd2/v_d_id20,v_d_id20,v_d_id21,v_d_id21
external roaddbtn,roadpart,riverdbtn,riverpart
parameter (title = 'Roads')
if(keybuf .lt. 0) then
   call uis$delete_window(v_d_id9)
call create_window1(v_d_id20,v_d_id20,-1.0,-1.0,9.0,3.0,10.0,4.0)
call uis$set_color(v_d_id20,0.9,1.0,0.0,0.0)
call uis$set_color(v_d_id20,1.0,0.0,0.0)
call uis$set_writing_index(v_d_id20,0.9,1)
call uis$text(v_d_id20,9,1,'Layout of Major Roads',0.0,2.50)
call uis$text(v_d_id20,9,2,'Particular Road',0.0,1.50)
call uis$text(v_d_id20,9,3,'Layout of Rivers & canals',0.0,0.50)
call uis$text(v_d_id20,9,4,'Particular River',0.0,-0.50)
call uis$button_astype(v_d_id20,v_d_id20,roaddbtn,.button,0.0,2.0,9.0,2.50)
call uis$button_astype(v_d_id20,v_d_id20,roadpart,.button,0.0,1.0,9.0,1.50)
call uis$button_astype(v_d_id20,v_d_id20,riverdbtn,.button,0.0,0.0,9.0,0.50)
call uis$button_astype(v_d_id20,v_d_id20,riverpart,.button,0.0,-1.0,9.0,-0.50)
end if
return
end

SUBROUTINE roadbtn( )
implicit integer (a-z)
include 'sys$library:uisentry'
include 'sys$library:uisusrdef'
common /cb1/v_d_id1,v_d_id2,v_d_id2,v_d_id5,v_d_id5,keybuf
common /cb2/button.keycode
common /vd2/v_d_id20,v_d_id20,v_d_id21,v_d_id21
common /os/ovlswitch
external window.priority,rd.util
if(button .lt. 0) then
call uis$delete_window(v_d_id20)
call ovlswitch(.ne. 1)
end if
return
end
 SUBROUTINE priorityrd( )

 Display the priority menu
 implicit integer (a-z)
 include 'sysLibrary:uisentry'
 include 'sysLibrary:uisusrdef'
 integer dsswitch,prn,cnt
 character(*) title
 common /cl/vd_id1.wd_id1,vd_id2.wd_id2,vd_id3.wd_id3,vd_id4.wd_id4,vd_id5.wd_id5,prn
 common /cl/vd_id1.wd_id1,vd_id2.wd_id2,vd_id3.wd_id3,vd_id4.wd_id4,vd_id5.wd_id5,prn
 common /cl/vd_id1.wd_id1,vd_id2.wd_id2,vd_id3.wd_id3,vd_id4.wd_id4,vd_id5.wd_id5,prn
 common /cl/vd_id1.wd_id1,vd_id2.wd_id2,vd_id3.wd_id3,vd_id4.wd_id4,vd_id5.wd_id5,prn
 parameter (title = 'PRIORITY MENU')

 call create_window1(vd_idp.wd_idp.0,0.0.0.5, 0.4, 0.4, 0.4, 0.4, 0.0, 0.0)

 call uis$set_color(vd_idp.0.1.0.1.0.0.0)
 call uis$set_color(vd_idp.1.1.0.0.0.0.0)
 call uis$set_color(vd_idp.0.9.1.0.0)
 call uis$set_color(vd_idp.1.1.0.0.0.0.0)
 call uis$set_color(vd_idp.0.9.1.0.0)
 call uis$set_color(vd_idp.1.1.0.0.0.0.0)
 call uis$set_color(vd_idp.0.1.0.1.0.0.0)
 call uis$set_color(vd_idp.1.1.0.0.0.0.0)
 call uis$set_color(vd_idp.0.9.1.0.0)
 call uis$set_color(vd_idp.1.1.0.0.0.0.0)
 call uis$set_color(vd_idp.0.1.0.1.0.0.0)
 call uis$set_color(vd_idp.1.1.0.0.0.0.0)
 call uis$set_color(vd_idp.0.9.1.0.0)
 call uis$set_color(vd_idp.1.1.0.0.0.0.0)
 call uis$set_color(vd_idp.0.1.0.1.0.0.0)
 call uis$set_color(vd_idp.1.1.0.0.0.0.0)
 call uis$set_color(vd_idp.0.9.1.0.0)
 call uis$set_color(vd_idp.1.1.0.0.0.0.0)
 call uis$set_button_ast(vd_idp.wd_idp.priorityrd1..keybuf.0.0.0.5, 0.4, 0.4, 0.4, 0.0)
 call uis$set_button_ast(vd_idp.wd_idp.priorityrd2..keybuf.0.0.0.5, 0.4, 0.4, 0.4, 0.0)
 call uis$set_button_ast(vd_idp.wd_idp.priorityrd3..keybuf.0.0.0.5, 0.4, 0.4, 0.4, 0.0)
 call uis$set_button_ast(vd_idp.wd_idp.priorityrd4..keybuf.0.0.0.5, 0.4, 0.4, 0.4, 0.0)
 call uis$set_button_ast(vd_idp.wd_idp.stptxt.-%ref(%loc(vd_idp)).

 return
 end

 SUBROUTINE priorityrd1( )

 Read with priority 1
 implicit integer (a-z)
 include 'sysLibrary:uisentry'
 include 'sysLibrary:uisusrdef'
 integer dsswitch,prn,cnt
 common /cl/vd_id1.wd_id1,vd_id2.wd_id2,vd_id3.wd_id3,vd_id4.wd_id4,vd_id5.wd_id5,prn
 parameter (title = 'PRIORITY MENU')

 if (keybuf .eq. uis$e_pointer_button_1) then
 prn = 1
 call entdbtnrd
 end if
SUBROUTINE entdbtnrd()

Sort out priorities and necessary parameters for transferring controls to knowledge base for searching.

implicit integer (a-z)
include 'sys$library:uisentry'
include 'sys$library:uisusrdef'
include ops$library:opsdef.for'
integer dswitch,prn,cnt
character*10 name1,name2,cname1

common /cb1/vd_id1,wd_id1,wd_id2,wd_id2,wd_id5,wd_id5,keybuf
external opsdbtn

if (prn .eq. 1) then
    name1 = 'ALROAD'
else if (prn .eq. 2) then
    name1 = 'ALROADM'
else if (prn .eq. 3) then
    name1 = 'ALROADR'
else if (prn .eq. 4) then
    name1 = 'ALROADO'
end if

name1 = 'RD_NAME'
name2 = 'NAME'
call opsdbtn(cname1, name1, name2)
return
end

Program fragment that activates the OPS5 production system after making necessary data type conversion suitable for OPS5.

SUBROUTINE opsdbtn(cname1, name1, name2)

implicit integer (a-z)
include 'sys$library:uisentry'
include 'sys$library:uisusrdef'
include ops$library:opsdef.for'
integer wme_cname, wme_name1, wme_name2, length, ln, i
integer naray(30)
character*10 came1, name1, name2
character*40 laray(30)
character*2 naray(30)
logical*1 keybuf(4)
logical completed
common /cb1/vd_id1, wd_id1, wd_id2, wd_id2, wd_id5, wd_id5, keybuf
common /vd2/vd_20, wd_id20, wd_id21, wd_id21
common /kb/kd_id, vcm_id1
common /os/osvswitch
common /le/count
common /la/laray, naray, naray
common /shm/vd_idd, wd_idd, cindex
external overlay.legend
cindex = cindex + 1
count = 1
do l = 1, 20
  laray(i) =
  naray(i) =
  narayn(i) = 0
end do
call ops$initialize()
call ops$clear
ln = length(cname1)
call ops$reset()
me_cname = ops$intern(%ref(cname1), %val(ln))
call ops$tab (%val(ops$cvna(%val(1))))
call ops$value (%val(ops$cname))
ln = length(name1)
wmec_name1 = ops$intern(%ref(name1), %val(ln))
call ops$tab (%val(ops$cvna(%val(2))))
call ops$value (%val(ops$cname1))
ln = length(name2)
wmec_name2 = ops$intern(%ref(name2), %val(ln))
call ops$tab (%val(ops$cvna(%val(2))))
call ops$value (%val(ops$cname2))
call ops$assert()
call opsstartup
call ops$run
call legend
call overlay
return
cend
C. 6. Creating A Control Element in the Working Memory

The control element in the production find-roads-rivers-rail, used for the query shown in 7.1 Plate 1, may be constructed as explained below. The control element in this production is (airdr1 ^triname name). Let the three strings be assigned to three string variables as in the following:

\[
\begin{align*}
\text{cnamel} &= \text{'airdr1'} \\
\text{name1} &= \text{'triname'} \\
\text{name2} &= \text{'name'} \\
\text{call ops$reset()} \\
\end{align*}
\]

This action clears the buffer called the result element which is used by VAX OPS5 run-time system to place the atoms before it is finally transferred to the working memory.

\[
\begin{align*}
\text{ln} &= \text{length (cnamel)} \\
\text{wme-cname} &= \text{ops$intern(%ref(cnamel), %val(ln))} \\
\text{call ops$tab(%val(OPS$cvna(%val(1))))} \\
\text{call ops$value(%val(wme-cname))} \\
\text{ln} &= \text{length(name1)} \\
\text{wme-name1} &= \text{ops$intern(%ref(name1),%val(ln))} \\
\text{call ops$tab(%val(ops$cvna(%val(2))))} \\
\text{call ops$value(%val(wme-name1))} \\
\text{ln} &= \text{length (name2)} \\
\text{wme-name2} &= \text{ops$intern(%ref(name2),%val(ln))} \\
\text{call ops$tab(%val(ops$cvna(%val(3))))} \\
\text{call ops$value(%val(wme-name2))} \\
\text{call ops$assert ()} \\
\end{align*}
\]

In the above, wme-cname, wme-name1, and wme-name2 are integer variables. Another integer variable ln holds the length of a string returned by the function length. The support routine ops$intern translates the character strings such as cnamel, name1, and name2 into symbols. The arguments to the function is the address of the string and the length of the string.
Another support routine, \texttt{opsScvna} converts an integer to an integer atom. The arguments for this routine such as integer values 1, 2, 3, \textit{et cetera}, indicate the corresponding field number in the result element into which the atom is to be placed.

The routine \texttt{opsTab} specifies the field in the result element in which the next entry is to be placed.

The role of \texttt{opsValue} routine is to place an atom in the result element.

Finally, \texttt{opsSassert} copies the contents of the result element into the working memory.
C. 7. Data Type Conversion Procedure While Invoking Within the OPS5 Environment, an External Procedure Written in a Language Different From OPS5

Consider the procedure call in the production **find-tourist-places-routes**, shown in 7.2, Plate 2.

```
(call clgplotall <name1> <location>
   <x-cord1> <y-cord1>
   <x-cord2> <y-cord2>)
```

The arguments in the procedure call *viz.*, `<name1>`, `<location>`, `<x-cord1>`, `<y-cord1>`, `<x-cord2>`, and `<y-cord2>`, are all symbolic atoms.

In the procedure `clgplotall` following type declarations have been made.

- `ename` and ` lname` character type variables of length 40
- `ename1`, ` lname1`, `ex-cord1`, `lx-cord1`, `ey-cord1`, and `ly-cord1` integer type variables
- `ex-cord`, `lx-cord`, `ey-cord`, and `ly-cord` real type variables

Now we have

```
c = ops$parametercount( )
enamel = ops$parameter (%val(1))
call ops$ pname (%val(enamel), %ref(enamel), %val(40))
lnamel = ops$parameter (%val(2))
call ops$ pname (%val(lnamel), %ref(lnamel), %val(40))
ex-cord1 = ops$parameter(%val(3))
if (ops$integer (%val(ex-cord1)))
   then ex-cord = ops$ cvan (%val(ex-cord1))
else if (ops$floating (%val(ex-cord1)))
   then ex-cord = ops$ cvaf (%val(ex-cord1))
   ey-cord1 = ops$parameter (%val(4))
```
if (ops$integer (%val(ey-cord1)))
    then ey-cord = ops$cvan(%val(ey-cord1))
else if (ops$floating (%val(ey-cord1)))
    then ey-cord = ops$cvaf (%val(ey-cord1))

This conversion process should be applied to other arguments also.

The ops$parametercount routine returns an integer that indicates the number of argument values stored in the result element. And the ops$parameter routine retrieves an argument value stored in the result element. This routine is to be specified with an integer that indicates the field from which the argument value is to be retrieved.

The ops$pname now translates the symbolic atom stored in the variable ename1 into a character string and copies it to the buffer ename. This routine is to be specified with three arguments, viz., the symbolic atoms to be translated, the address of the buffer to which the string is to be copied, and the integer that represents the number of characters in the string.

The ops$integer routine tests whether or not an atom is an integer, and returns a boolean result. Similarly, ops$floating test whether or not an atom is a floating point number and returns a boolean result.

The routine ops$cvan converts an integer atom into an integer and returns the result. Similarly, ops$cvaf takes a floating point argument and converts it into a floating point number.
Appendix D

D. 1. Definition of the Production find-all-colleges-NH47
    (vide 7.5. Plate 5)
    (p find-all-colleges-NH47
     {<edn> (alclg ^name name1})
     (road ^rd-name [national-highway-47]
      ^rd-cord <rd-cord>)}
     {<clgs> (edn-research
      ^type college
      ^name <name>
      ^location <location>
      ^x-cord <x-cord1>
      ^y-cord <y-cord1>
      ^clg-no <clg-no>
      ^criteria (adjacent <rd-cord>))}
     -->
     (call clgplotall <name> <x-cord1> <y-cord1> <clg-no>))

D. 2. Definition of the Production find-all-hospital
    (vide 7.6. Plate 6)
    (p find-all-hospital
     {<alhosp> (ahosp ^name <name1>)}
     (hospital ^name <name>
      ^location <location>
      ^x-cord <x-cord1>
      ^y-cord <y-cord1>
      ^h-no <hno>)}
     (place ^pname <location>
      ^x-cord <x-cord2>
      ^y-cord <y-cord2>)
     -->
     (call clgplotall <name> <x-cord1> <y-cord1> <hno>))
D. 3. Definition of the Production **find-schools-around-secretariat**
(vide 7.7 Plate 7)

\[
\begin{align*}
(p & \text{find-schools-around-secretariat} \\
& \text{(schlsec}} \quad \text{\textasciitilde \text{name}} \quad \text{\textless \text{name1\textgreater}}) \\
& \text{(office-building} \quad \text{\textasciitilde \text{type}} \quad \text{government} \\
& \quad \text{\textasciitilde \text{name}} \quad \text{secretariat} \\
& \quad \text{\textasciitilde \text{x-cord}} \quad \text{\textless \text{x-cord1\textgreater}} \\
& \quad \text{\textasciitilde \text{y-cord}} \quad \text{\textless \text{y-cord1\textgreater}}) \\
& \text{(edn-research} \quad \text{\textasciitilde \text{type}} \quad \text{school} \\
& \quad \text{\textasciitilde \text{name}} \quad \text{\textless \text{name\textgreater}} \\
& \quad \text{\textasciitilde \text{location}} \quad \text{\textless \text{location\textgreater}} \\
& \quad \text{\textasciitilde \text{x-cord}} \quad \text{\textless \text{x-cord2\textgreater}} \\
& \quad \text{\textasciitilde \text{y-cord}} \quad \text{\textless \text{y-cord2\textgreater}} \\
& \quad \text{\textasciitilde \text{schl-no}} \quad \text{\textless \text{schl-no\textgreater}} \\
& \quad \text{\textasciitilde \text{criteria}} \quad \text{\textless \text{between \textless \text{2\textgreater} \quad \textless \text{6\textgreater} \quad \textless \text{secretariat\textgreater\textgreater})} \\
& \text{(place} \quad \text{\textasciitilde \text{pname}} \quad \text{\textless \text{location\textgreater}} \\
& \quad \text{\textasciitilde \text{x-cord}} \quad \text{\textless \text{x-cord3\textgreater}} \\
& \quad \text{\textasciitilde \text{y-cord}} \quad \text{\textless \text{y-cord3\textgreater})} \\
\longrightarrow\end{align*}
\]

\[
\begin{align*}
& \text{(call clgplot} \quad \text{\textless \text{name\textgreater}} \quad \text{\textless \text{x-cord2\textgreater}} \\
& \quad \text{\textless \text{y-cord2\textgreater} \quad \text{\textless \text{schl-no\textgreater\textgreater})}
\end{align*}
\]