Alvey MMI-007 Vehicle Exemplar: Object Hypothesis by Evidential Reasoning

S. K. Morton

Information Technology Research Centre,
University of Bristol, BS8 1TR.

ABSTRACT

Firstly the need for an approach to the design of an integrated vision system incorporating both bottom-up and top-down techniques is explained. Secondly two theories that may be applied in such a system, namely fuzzy conceptual schemas and support logic, are expounded. Thirdly a program utilising such tools and envisaged as a prototype generalised vision system is described. Fourthly results obtained for a restricted domain of images are commented upon.

INTRODUCTION

Previous work on artificial vision can be roughly divided into three camps, which may be characterised by the terms "neuronal", "psycho-physical", and "knowledge-based".

The neuronal approach is epitomised by the work of Aleksander (Aleksander & Stonham 1979) and by the more recent efforts of the connectionist school (e.g. Hinton & Anderson 1981), where a network of artificial pseudo-neurons - random-access-memories in Aleksander's work, processors in the connectionists' work - "learns" representations of objects as states of the network. Impressive results have been obtained, and an especially interesting feature in such an approach is the ability to recognise an object with a minor flaw as the object previously learnt but with a diminished confidence, a characteristic of natural vision.

The psychophysical approach is based on the philosophy of Artificial Intelligence (AI) propounded by Marr (1977, 1982). Here the emphasis is on the establishment of "hard" theories in the physics of visual processes which can be built upon by subsequent research. Examples are the difference-of-Gaussians edge-detection process and the 2.5D sketch technique devised by Marr (1982). The procedures involved are of the deterministic mathematical paradigm, and there is no attempt to incorporate representations of the higher-level processes of human vision, particularly symbolic inference.

The approach of knowledge-based methods however is founded principally on the notion that human vision consists of the verification of hypotheses arising from the expectations embodied in background knowledge of possible situations in the world. Thus symbolic reasoning is viewed as the guide for the invocation of lower-level processes, and a greater experience of visual contexts implies a greater visual ability.

It seems clear that none of the above on their own will provide a satisfactory solution to the vision problem. The neuronal and psychophysical approaches appear to be effective for the recognition of isolated objects from restricted domains, but not so for general scene interpretation where a number of objects may be juxtaposed with background areas undefinable in terms of shape properties. On the other hand the knowledge-based approach addresses the principle that context has an important role to play but it is apparent that something more is needed.

The approach taken in this paper is that the interpretation of a scene may only be performed by the bringing of various sorts of knowledge to bear on the raw data using a combination of data-driven and knowledge-driven techniques. Central to this is the need to combine the evidences arising from different sources, which, due to the imperfections of the segmented data, may be partial evidences. Therefore some well-founded calculus must be employed to take account of such uncertainty in the conflation of various supportive and counter-supportive evidences. By this approach the complex interactions of human vision may begin to be modelled. Until the fuller development of parallel processing methodologies and the results of connectionist models become more well established, it is only possible to develop large programs for vision using serial languages; however, important results emerge which may be relevant to implementations on more suitable hardware.

Thus the purpose of this paper is to emphasise the need for a greater interaction between the various approaches in artificial vision, and in a later section a program embodying a prototype of the sort of system architecture required is presented. Firstly however two theoretical tools utilised in this program are described.

I FUZZY CONCEPTUAL SCHEMAS

A knowledge representation formalism expounded by Sowa (1984) is the theory of conceptual graphs. A conceptual graph consists of concepts linked via conceptual relations: concepts correspond to entities in the world while conceptual relations express the semantic relations holding between these entities. Conceptual graphs play a part in the vision program described below in two ways: as perceptual graphs, symbolic representations of the scene's interpretation and as schemas, incorporating background knowledge. Perceptual graphs are described in a later section.

A schema is a kind of conceptual graph embodying knowledge in the form of expectations and associations for a particular concept, the subject of the schema. It is a declarative knowledge structure and may be used in non-deductive reasoning. In the context of vision the concepts and conceptual relations refer to specifically visual concepts and relations: thus no event concepts and no verb-noun case relations are included, the assumption being that this type of knowledge is only very indirectly conveyed by visual processes. The types of the concepts thus refer to expressible physical entities and directly perceptible attributes, while the relations refer to entity-attribute relationships and perceptible two-dimensional semantic constraints and expectations holding between entities.

AVC 1987 doi:10.5244/C.1.3

A fuzzy conceptual schema is a schema for which conditional support pairs (see next section) are associated with conjunctions of the relations of the schema expressing their contributory evidential supports for and against the subject. In Figure 1 an example of a visual fuzzy conceptual schema for the concept [building] is shown. It consists of three conceptual links representing the relationships "a roof is a part of a building", "a walling is a part of a building", "a roof is above a walling". These conceptual relations pertain to the image domain rather than to the object domain, i.e. the area of the image corresponding to "building" subsumes the area of the image corresponding to "roof", etc. These relationships constitute the expected associations for an image of a building. The relative importances of conjunctions of these conceptual links is given by their attached support pairs; for more explanation of this type of knowledge structure see Morton & Popham (1987).

```
schema(building,*x4) <~
  [building:*x4] -
    (*r1,part) --> [roof] -
      (*r3,above) <-- [walling:*x6];
    (*r2,part) --> [walling:*x6];
  {{*r1}:[1,1],{*r2}:[1,1],
   {*,*}:[1,1],{*r1,*r2,*r3}:[1,1]}.
```

Figure 1

N.B. The notation {*,*}:[x,y] means that all conjunctions with two members have the support pair [x,y].

II SUPPORT LOGIC

The calculus that has been used in the development of the vision program herein described is that of support logic (Baldwin 1985) which has been developed at Bristol University and implemented as SLOP (Monk & Baldwin 1987). Support logic is a PROLOG-like programming language which incorporates uncertainty. A support logic program consists of facts, rules and bundles. A fact is of the form of a PROLOG fact with an associated support pair $[\alpha,\beta]$, $\alpha,\beta \ \epsilon [0,1]$, where α denotes the degree of necessary support for the fact, and $1-\beta$ the degree of necessary support for the negation of the fact. A rule is of the form of a PROLOG rule with an associated conditional support pair $[\alpha,\beta]$, $\alpha,\beta \ \epsilon [0,1]$, where α denotes the degree of necessary support for the head of the rule given the body, and $1-\beta$ the degree of necessary support for the negation of the head given the body. A bundle is a compound rule with one head and a number of bodies with their own conditional support pairs used to represent mutually dependent rules. Figure 2 shows a bundle for the predicate trees.

The calculus is founded on ideas from fuzzy logic and Dempster's evidence theory and has been devised to combine and propagate values for supports through a chain of inference involving support logic facts, rules and bundles (Baldwin 1986). This calculus has been detailed elsewhere (Baldwin & Monk 1987), (Monk & Baldwin 1987), (Morton & Popham 1987).

```
trees(X):-
        <-
            green(Y),
            has_hue(Y,X) :[0.6,1]
        <-
            value(Y@$low_val),
            has_val(Y@$low_val,X) :[0.6,1]
        <-
            sky(Y),
            adj(Y,X) :[0.3,1]
        <-
            green(Y),
            has_hue(Y,X),
            sky(Z),
            adj(Z,X) :[0.75,1]
        <-
            green(Y),
            has_hue(Y,X),
            value(Z@$low_val),
            has_val(Z@$low_val,X) :[0.9,1]
        <-
            value(Y@$low_val),
            has_val(Y@$low_val,X),
            sky(Z),
            adj(Z,X) :[0.85,1]
        <-
            green(Y),
            has_hue(Y,X),
            value(Z@$low_val),
            has_val(Z@$low_val,X),
            sky(W),
            adj(W,X) :[1,1].
```

Figure 2

It is worth describing an example of how the SLOP system interfaces with the data. In a typical stage of the program it may be necessary to evaluate the "greenness" of a set of contiguous regions from the segmentation, say [1,2,3]. Then we invoke the rules below:-

```
area_has_hue(Hue,A,Size) :-
    call(member(X,A)),
    has_hue_proc(Hue,X),
    relatively_big_region(X,Size) :[1,1].

area_has_hue(Hue,A,Size) :-
    call(member(X,A)),
    sup_not has_hue_proc(Hue,X),
    relatively_big_region(X,Size) :[0,0].

has_hue_proc(green,X) :-
    call(colour(X,Hue,Sat)),
    call(deg_to_rad(Hue,HueRad)),
    call(colour_compatible(HueRad,Sat,1.0472,3.14159,
                                  N,P)) :[N,P].

relatively_big_region(X,Size) :-
    call(geometry(X,_,_,XSize,_,_,_,_)),
    call(N is sqrt(XSize/Size)) :[N,N].
```

To explain these predicates the reasoning to answer the SLOP query is described.

```
area_has_hue(green,[1,2,3],150). ,
```

This asks for the support that the set of regions or area |1,2,3| has the hue green, when the aggregate size of the regions is 150 (pixels). Resolving this with the first area_has_hue clause gives us the positive support provided by relatively_big_regions (compared to 150) which are green; resolving with the second clause gives us the negative support provided by the relatively_big_regions which are sup_not (not) green. Obviously support pairs are returned from the relatively_big_region and has_hue_proc clauses, and these are combined according to the support logic calculus. The above query can be satisfied by each region number twice, once positively, once negatively so that we would have six contributing support pairs; these would be combined according to the renormalisation rule of the calculus. The trace of the above query is shown in Appendix I.

The has_hue_proc clause and the relatively_big_region clause call the colour predicate and the geometry predicate for the relevant region respectively. These embody part of the actual data evaluated during the segmentation process. (N.B. Ordinary PROLOG clauses are called from SLOP using the "call" predicate.)

In Appendix II there is a trace of the evaluation of the trees bundle shown in Figure 2. The bundle has seven rules comprising conjunctions of the conceptual links in the corresponding schema together with their associated conditional support pair. These are evaluated one at a time and the resulting support pairs are treated as intervals so that their intersection becomes the overall support for trees ([4,5,6]).

III THE VISION PROGRAM: GLAICIER

GLAICIER is an acronym for General Analysis of Images using Contextual Information and Evidential Reasoning. It is intended to be a prototype general scene analyser where there is an integration of knowledge-based methods and lower-level processes and techniques. Interaction occurs in both directions: low-level evaluators provide clues or stimuli to high-level knowledge structures; and the resulting hypotheses are verified by the application of other evaluators to the image data. This sort of mutual interaction occurs between different levels of representation, and is an attempt to model theories of such interaction in the human vision system.

The working of the program is much as described in Morton & Popham(1987), so there is little point in going through all the implementation details again. Instead a different approach is taken: the various different levels of representation and their mutual interactions are considered.

At the moment the lowest level the program deals with is the segmentation level; this means that it has no interaction with the pixel level itself, and so we are unable to revise the segmentation if it seems necessary. These data come from MCCS in an array format, and is converted into a PROLOG representation using a specifically written C program.

The second level is what is termed the plan, a chunking together of regions using topological data and colour information to resolve conflicts. This is represented as a number of PROLOG clauses containing the dominant region and subsidiary regions for each plan area.

The third level is what is termed the plan_image where each plan area has been assigned an ordered list of possible labels based on the evaluation of certain clues (see below) and of a number of properties drawn from the schemas.

The next level is that of perceptions which are the result of the application of the schemas and evaluation of all the accumulated evidence using the support logic calculus. They are in fact instantiations of the concepts in the schemas with support pairs associated with the label assignations to regions or groups of regions.

The final level is the perceptual graph which is the symbolic description of the image in the format of a conceptual graph derived by joining together the perceptions representing physobj (physical object) types. In other words attribute concepts that have been established are ignored for the sake of clarity.

The first interaction consists of the application of low-level clue processes to the regions which, if resulting in a positive outcome, leads to applying the schema corresponding to the clue to the subsumptive plan area. For example, there are several profile features which strongly indicate the presence of a car; if these are incorporated as clues, then a positive outcome to their application to a region causes the schema for road to be subsequently applied to that region's plan area as a priority.

The second interaction consists of the application of a schema to the plan image. This causes the invocation of low-level region feature and relation evaluators and their application to the segmentation data for a group of contiguous regions in the image. The subject of the schema may be a decomposable concept; that is, it subsumes other concepts, in which case other schemas are invoked. For certain highly structured concepts a schema is not appropriate; rather a model-matching process using a model of the object in question is applied to an area of the image suggested by the schema. The model-matching process itself is a low-level 6-D template-matching procedure which has been implemented at Reading University. (Since they have the original images, they are able to apply it directly to the pixel data.) This is an expensive process, so that the advantage of the integration of the top-down approach with geometrical model-matching technique is apparent in terms of efficiency.

Investigations are currently under way to devise some way of assessing the plan, the structure of which is crucial, and thus reconstructing it if necessary. This assessment can be done when we have "bad" shapes, "bad" labels or "bad" aggregations of regions, which can be rectified by reassignment of subsidiary regions, merging of regions or by using a greater resolution in problematic localities in the image.

IV RESULTS

Figures 3 to 7 show the successive representations of an image when GLAICIER is applied to it. Thus the digitised image in Figure 3 is ultimately interpreted as the perceptual graph in Figure 7. This particular image is a member of the Alvey image set. It was chosen since it is a typical example of the type of scene to which GLAICIER is designed to be applicable and moreover includes an unambiguous car.

In Figure 4 is the MCCS segmentation of the image with the boundaries of the plan areas shown in bold. 'I' identifies the image; 'R' identifies which of the five segmentations corresponding to different resolutions is used (in this case the lowest resolution); and 'P' is the parameter used in the formation of the plan. Obviously the resolution and parameter have been chosen to give a "good" plan; as previously mentioned it is desirable to incorporate some means by which assessment and selection of the plans can be performed automatically.

In Figure 5 are the plan image areas which result from the application of certain attribute evaluators to the plan areas. For the types building and field no simple attributes are pertinent, so no information is gained about these labels at this stage; hence the [0,1] support pairs. The other support pairs are not recorded; they are merely used to give an ordering to the schemas to be applied to each plan image area.

In Figure 6 is the list of percepts which result from the application of the schemas. The structure of the image, as expressed by the plan, is represented by a number of part relationships. Furthermore the structure of a plan area deemed to correspond to a decomposable concept is likewise represented. Regions and groups of regions are given entity labels. A trace of the process of evaluation is shown in Appendix III; note that interrogation of the operator is used to elucidate the supports for and against a car being in a certain area, since the model-matching process is not fully integrated.

In Figure 7 is the resulting fuzzy perceptual graph as the symbolic representation of the image. The syntax is the same as Sowa's for ordinary conceptual graphs, except that within each concept there is a support pair incorporated delimited by a colon. Perceptual relations are not ascribed support pairs.

V CONCLUSIONS

Clearly the resulting perceptual graph of the last section is only a rough representation of the image and in the case of regions 25 and 24 completely inaccurate. The reason for this is that since the interpretation relies exclusively on the data resulting from the MCCS segmentation, any assessment of region properties and relationships in the evaluation of schemas will only be as semantically valid as the data. For example regions corresponding to areas of the image which are obviously green may not necessarily be assigned the "correct" colour values. Thus these regions' assessment as trees or grass will be adversely affected. Accuracy of colour identification is well-known as problematic; some sort of normalisation of the values may be helpful.

Improvement in the system performance may be obtained if other information about the image is used, in particular edge-maps. Curves, straight lines, closed loops and contiguous groups thereof may provide important clues to certain types of object, particularly artefacts.

Thus a source of evidence independent of the segmentation data could be used in tandem with that data. Another way in which extra information can be used is if the five MCCS resolutions are arranged in an hierarchical format. This would enable the analysis to be performed with greatest detail at difficult areas of the image while using the lowest resolution for homogeneous "uninteresting" areas such as sky.

Due to the system's total dependence on the segmentation data, its results are not always accurate. Experimentation is proceeding with other images in the Alvey set and adaptations and extensions to the programs are continually carried out in order to improve performance and increase generality. It is not worth pretending that there is not a long way to go before we have a general scene analysis program; but at least we have started.

REFERENCES

1 Aleksander, I., & Stonham, T.J., "Guide to pattern recognition using random-access memories", I.E.E. Journal on Computers and Digital Techniques, II.1, (1979), 29-40.

2 Hinton, G.E., & Anderson, J.A., Parallel Models of Associative Memory, Lawrence Erlbaum Associates, New York, New York, U.S.A., 1981.

3 Marr, D., "Artificial Intelligence - a personal view", Artificial Intelligence, IX.1, (1977), 37-48.

4 Marr, D., Vision, W.H. Freeman, San Francisco, California, U.S.A., 1982.

5 Sowa, J.F., Conceptual Structures, Addison-Wesley, 1984.

6 Morton, S.K., & Popham, S.J., "An algorithm design specification for interpreting segmented image data using schemas and support logic", Information Technology Research Centre Report No. ITRC 85, University of Bristol, Bristol, U.K. (Alvey Ref.No. AOI/PR/BU/860922; to be published in Image and Vision Computing, August 1987).

7 Baldwin, J.F., "Support Logic Programming", Proc. NATO Advanced Study Institute on Fuzzy Sets, 8th-20th July 1985.

8 Monk, M.R.M., & Baldwin, J.F., "Slop User's Manual Version 1.2", Information Technology Research Centre Report No. 106, University of Bristol, Bristol, U.K., 1987.

9 Baldwin, J.F., "Evidential Support Logic Programming", Information Technology Research Centre Report No. 80, University of Bristol, Bristol, U.K., 1986.

10 Baldwin, J.F., & Monk, M.R.M., "Evidence Theory Fuzzy Logic and Logic Programming", Information Technology Research Centre Report No. 109, University of Bristol, Bristol, U.K., 1987.

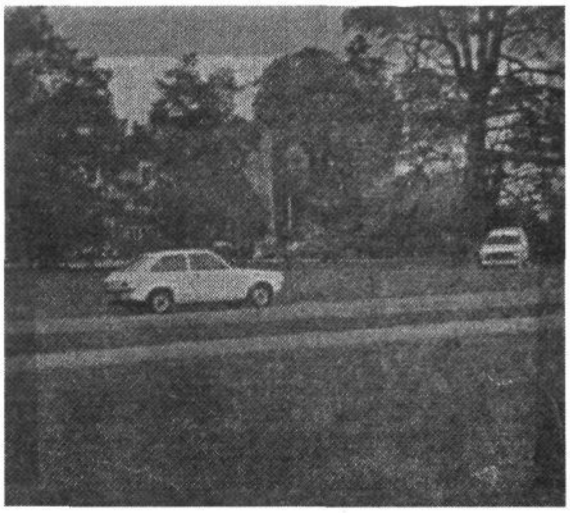
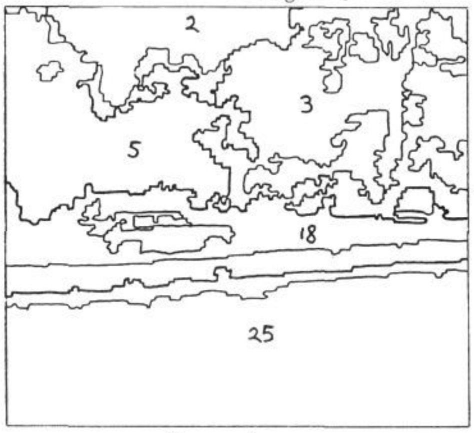


Figure 3



P = 20
R = 1
I = ru02

Figure 4

```
plan_area(1,2,[11,13]).
plan_area(1,3,[4,6,7,8,9,10,14,15,16,17]).
plan_area(1,5,[1,12]).
plan_area(1,18,[19,20,21,22,23]).
plan_area(1,25,[24]).
```

```
plan_image_area(1,2,[11,13],
                [(sky:[0.811793,0.902015]),
                 (road:[0.0819567,0.263078]),
                 (building:[0,1]),
                 (trees:[0.104003,0.857502]),
                 (field:[0,1])]).

plan_image_area(1,3,[4,6,7,8,9,10,14,15,16,17],
                [(sky:[0.276075,0.364774]),
                 (road:[0,0.0297936]),
                 (building:[0,1]),
                 (trees:[0.842552,0.908525]),
                 (field:[0,1])]).

plan_image_area(1,5,[1,12],
                [(sky:[0.0024077,0.0165514]),
                 (road:[0,0.0121156]),
                 (building:[0,1]),
                 (trees:[0.98001,0.995545]),
                 (field:[0,1])]).

plan_image_area(1,18,[19,20,21,22,23],
                [(sky:[0,0.0512509]),
                 (road:[0.915675,0.981105]),
                 (building:[0,1]),
                 (trees:[0.709622,0.826977]),
                 (field:[0,1])]).

plan_image_area(1,25,[24],
                [(sky:[0,0.0258656]),
                 (road:[0.114809,0.296122]),
                 (building:[0,1]),
                 (trees:[0.958712,0.991509]),
                 (field:[0,1])]).
```

Figure 5

```
percept(image(ru02),[1,1]).
percept(part([2,11,13],ru02),[1,1]).
percept(part([3,4,6,7,8,9,10,14,15,16,17],ru02),[1,1]).
percept(part([5,1,12],ru02),[1,1]).
percept(part([18,19,20,21,22,23],ru02),[1,1]).
percept(part([25,24],ru02),[1,1]).
percept(part([2],[2,11,13]),[1,1]).
percept(part([11],[2,11,13]),[1,1]).
percept(part([13],[2,11,13]),[1,1]).
percept(cloud([2]),[0.532657,1]).
percept(cloud([11]),[0.538266,1]).
percept(cloud([13]),[0.557345,1]).
percept(sky([2,11,13]),[0.90448,1]).
percept(adj([3,4,6,7,8,9,10,14,15,16,17],[2,11,13]),[1,1]).
percept(trees([3,4,6,7,8,9,10,14,15,16,17]),[0.800675,1]).
percept(adj([5,1,12],[2,11,13]),[1,1]).
percept(trees([5,1,12]),[0.8352,1]).
percept(part([18],[18,19,20,21,22,23]),[1,1]).
percept(part([22,20,21],[18,19,20,21,22,23]),[1,1]).
percept(part([23],[18,19,20,21,22,23]),[1,1]).
percept(part([19],[18,19,20,21,22,23]),[1,1]).
percept(tarmac([18]),[0.4,1]).
percept(tarmac([23]),[0.509547,1]).
percept(adj([25,24],[2,11,13]),[0,0]).
percept(trees([25,24]),[0.830072,1]).
percept(car([22,20,21]),[1,1]).
percept(car([19]),[0.9,0.9]).
percept(road([18,19,20,21,22,23]),[0.995079,1]).
```

Figure 6

```
[image:ru02:[1,1]] -
        (part) --> [sky:[2,11,13]:[0.90448,1]] -
                (part) --> [cloud:[2]:[0.532657,1]]
                (part) --> [cloud:[11]:[0.538266,1]]
                (part) --> [cloud:[13]:[0.557345,1]]
                (adj) --> [trees:[3,4,6,7,8,9,10,14,15,16,17]:[0.800675,1]]

                (adj) --> [trees:[5,1,12]:[0.8352,1]]
                (adj) --> [trees:[25,24]:[0.830072,1]];
        (part) --> [trees:[3,4,6,7,8,9,10,14,15,16,17]:[0.800675,1]]
        (part) --> [trees:[5,1,12]:[0.8352,1]]
        (part) --> [road:[18,19,20,21,22,23]:[0.995079,1]] -
                (part) --> [tarmac:[18]:[0.4,1]]
                (part) --> [car:[22,20,21]:[1,1]]
                (part) --> [tarmac:[23]:[0.509547,1]]
                (part) --> [car:[19]:[0.9,0.9]];
        (part) --> [trees:[25,24]:[0.830072,1]];
```

Figure 7

Appendix 1

```
| ?- slop.

Support Logic Programming - Version 1.1
J.F.Baldwin and M.R.M.Monk
I.T.R.C., Dept. of Engineering Mathematics,
University of Bristol, England.
January 1986


query? trace,area_has_hue(green,[1,2,3],150).

area_has_hue(green,[1,2,3],150):-
        call(member(_161,[1,2,3])),
        has_hue_proc(green,_161),
        relatively_big_region(_161,150) :[1,1].

OVERALL SUPPORT -> member(1,[1,2,3]) :[1,1]

->      has_hue_proc(green,1) :[0.7,0.8]
OVERALL SUPPORT -> has_hue_proc(green,1) :[0.7,0.8]

->      relatively_big_region(1,150) :[0.1,0.1]
OVERALL SUPPORT -> relatively_big_region(1,150) :[0.1,0.1]

->      call(member(1,[1,2,3])),
        has_hue_proc(green,1),
        relatively_big_region(1,150) :[0.0699999,0.0799999]
->      area_has_hue(green,[1,2,3],150) :[0.0699999,1]
OVERALL SUPPORT -> member(2,[1,2,3]) :[1,1]

->      has_hue_proc(green,2) :[0.9,0.95]
OVERALL SUPPORT -> has_hue_proc(green,2) :[0.9,0.95]

->      relatively_big_region(2,150) :[0.8,0.8]
OVERALL SUPPORT -> relatively_big_region(2,150) :[0.8,0.8]

->      call(member(2,[1,2,3])),
        has_hue_proc(green,2),
        relatively_big_region(2,150) :[0.719999,0.759999]
->      area_has_hue(green,[1,2,3],150) :[0.719999,1]
OVERALL SUPPORT -> member(3,[1,2,3]) :[1,1]

->      has_hue_proc(green,3) :[0.05,0.2]
OVERALL SUPPORT -> has_hue_proc(green,3) :[0.05,0.2]

->      relatively_big_region(3,150) :[0.9,0.9]
OVERALL SUPPORT -> relatively_big_region(3,150) :[0.9,0.9]

->      call(member(3,[1,2,3])),
        has_hue_proc(green,3),
        relatively_big_region(3,150) :[0.045,0.18]
->      area_has_hue(green,[1,2,3],150) :[0.045,1]
```

```
Call to Prolog Goal - call(member(_161,[1,2,3])) FAILED
area_has_hue(green,[1,2,3],150):-
        call(member(_161,[1,2,3])),
        sup_not has_hue_proc(green,_161),
        relatively_big_region(_161,150) :[0,0].

OVERALL SUPPORT -> member(1,[1,2,3]) :[1,1]

->      has_hue_proc(green,1) :[0.7,0.8]
OVERALL SUPPORT -> has_hue_proc(green,1) :[0.7,0.8]

->      sup_not has_hue_proc(green,1) :[0.2,0.3]
->      relatively_big_region(1,150) :[0.1,0.1]
OVERALL SUPPORT -> relatively_big_region(1,150) :[0.1,0.1]

->      call(member(1,[1,2,3])),
        sup_not has_hue_proc(green,1),
        relatively_big_region(1,150) :[0.02,0.03]
->      area_has_hue(green,[1,2,3],150) :[0,0.98]
OVERALL SUPPORT -> member(2,[1,2,3]) :[1,1]

->      has_hue_proc(green,2) :[0.9,0.95]
OVERALL SUPPORT -> has_hue_proc(green,2) :[0.9,0.95]

->      sup_not has_hue_proc(green,2) :[0.0500002,0.1]
->      relatively_big_region(2,150) :[0.8,0.8]
OVERALL SUPPORT -> relatively_big_region(2,150) :[0.8,0.8]

->      call(member(2,[1,2,3])),
        sup_not has_hue_proc(green,2),
        relatively_big_region(2,150) :[0.0400001,0.0800002]
->      area_has_hue(green,[1,2,3],150) :[0,0.96]
OVERALL SUPPORT -> member(3,[1,2,3]) :[1,1]

->      has_hue_proc(green,3) :[0.05,0.2]
OVERALL SUPPORT -> has_hue_proc(green,3) :[0.05,0.2]

->      sup_not has_hue_proc(green,3) :[0.8,0.95]
->      relatively_big_region(3,150) :[0.9,0.9]
OVERALL SUPPORT -> relatively_big_region(3,150) :[0.9,0.9]

->      call(member(3,[1,2,3])),
        sup_not has_hue_proc(green,3),
        relatively_big_region(3,150) :[0.719999,0.854999]
->      area_has_hue(green,[1,2,3],150) :[0,0.280001]
Call to Prolog Goal - call(member(_161,[1,2,3])) FAILED
OVERALL SUPPORT -> area_has_hue(green,[1,2,3],150) :[0.443162,0.589846]


area_has_hue(green,[1,2,3],150) :[0.443162,0.589846]
```

Appendix 2

```
| ?- slop.

Support Logic Programming - Version 1.1
J.F.Baldwin and M.R.M.Monk
I.T.R.C., Dept. of Engineering Mathematics,
University of Bristol, England.
January 1986


query? trace,trees([4,5,6]).
```

```
 trees([4,5,6]):-
          <-
                   green(_152),
                   has_hue(_152,[4,5,6]) :[0.6,1]
          <-
                   value(_153@$low_val),
                   has_val(_153@$low_val,[4,5,6]) :[0.6,1]
          <-
                   sky(_154),
                   adj(_154,[4,5,6]) :[0.3,1]
          <-
                   green(_155),
                   has_hue(_155,[4,5,6]),
                   sky(_156),
                   adj(_156,[4,5,6]) :[0.75,1]
          <-
                   green(_157),
                   has_hue(_157,[4,5,6]),
                   value(_158@$low_val),
                   has_val(_158@$low_val,[4,5,6]) :[0.9,1]
          <-
                   value(_159@$low_val),
                   has_val(_159@$low_val,[4,5,6]),
                   sky(_160),
                   adj(_160,[4,5,6]) :[0.85,1]
          <-
                   green(_161),
                   has_hue(_161,[4,5,6]),
                   value(_162@$low_val),
                   has_val(_162@$low_val,[4,5,6]),
                   sky(_163),
                   adj(_163,[4,5,6]) :[1,1].

->        <-
                   green(_152),
                   has_hue(_152,[4,5,6]) :[0.6,1] :[0.3,1]

->        <-
                   value(_153@$low_val),
                   has_val(_153@$low_val,[4,5,6]) :[0.6,1] :[0.539999,1]

->        <-
                   sky(_154),
                   adj(_154,[4,5,6]) :[0.3,1] :[0,1]


->        <-
                   green(_155),
                   has_hue(_155,[4,5,6]),
                   sky(_156),
                   adj(_156,[4,5,6]) :[0.75,1] :[0,1]

->        <-
                   green(_157),
                   has_hue(_157,[4,5,6]),
                   value(_158@$low_val),
                   has_val(_158@$low_val,[4,5,6]) :[0.9,1] :[0.404999,1]

->        <-
                   value(_159@$low_val),
                   has_val(_159@$low_val,[4,5,6]),
                   sky(_160),
                   adj(_160,[4,5,6]) :[0.85,1] :[0,1]

->        <-
                   green(_161),
                   has_hue(_161,[4,5,6]),
                   value(_162@$low_val),
                   has_val(_162@$low_val,[4,5,6]),
                   sky(_163),
                   adj(_163,[4,5,6]) :[1,1] :[0,1]
->        trees([4,5,6]) :[0.539999,1]
OVERALL SUPPORT -> trees([4,5,6]) :[0.539999,1]


trees([4,5,6]) :[0.539999,1]
```

Appendix 3

```
| ?- apply_schemas(1,0.8).

Applying schema for sky i.e.:

schema(sky,*x1) <~
        [sky:*x1] -
                (part) --> [cloud]
                (part) --> [blue_sky]
                (loc) --> [image_top];

to regions [2,11,13]

SLOP fact retrieved: loc(image_top,[2,11,13]) :[0.811793,0.902015]

Subplan formed for area [2,11,13] is

[(2,[]),(11,[]),(13,[])]

Applying schema for cloud i.e.:

schema(cloud,*x5) <~
        [cloud:*x5] -
                (has_hue) --> [grey]
                (has_val) --> [value@$high_val]
                (part) <-- [sky];

to regions [2]

SLOP fact determined: has_hue(grey,[2]) :[0.887762,0.988556]

SLOP fact determined: has_val(value@$high_val,[2]) :[0.100732,0.355524]

SLOP fact determined: cloud([2]) :[0.532657,1]

Applying schema for blue_sky i.e.:

schema(blue_sky,*x9) <~
        [blue_sky:*x9] -
                (has_hue) --> [blue]
                (has_val) --> [value@$high_val]
                (part) <-- [sky];

to regions [2]

SLOP fact determined: has_hue(blue,[2]) :[0,0]

SLOP fact determined: has_val(value@$high_val,[2]) :[0.100732,0.355524]

SLOP fact determined: blue_sky([2]) :[0.0906587,1]

Applying schema for cloud

to regions [11]

SLOP fact determined: has_hue(grey,[11]) :[0.89711,0.990316]

SLOP fact determined: has_val(value@$high_val,[11]) :[0,0]

SLOP fact determined: cloud([11]) :[0.538266,1]

Applying schema for blue_sky
to regions [11]

SLOP fact determined: has_hue(blue,[11]) :[0,0]

SLOP fact determined: has_val(value@$high_val,[11]) :[0,0]

SLOP fact determined: blue_sky([11]) :[0,1]

Applying schema for cloud
to regions [13]

SLOP fact determined: has_hue(grey,[13]) :[0.928909,0.99526]

SLOP fact determined: has_val(value@$high_val,[13]) :[0,0]

SLOP fact determined: cloud([13]) :[0.557345,1]

Applying schema for blue_sky
```

```
     to regions [13]

     SLOP fact determined: has_hue(blue,[13]) :[0,0]

     SLOP fact determined: has_val(value@$high_val,[13]) :[0,0]

     SLOP fact determined: blue_sky([13]) :[0,1]

     SLOP fact determined: sky([2,11,13]) :[0.90448,1]

     Applying schema for trees i.e.:

     schema(trees,*x29) <~
             [trees:*x29] -
                     (has_hue) --> [green]
                     (has_val) --> [value@$low_val]
                     (adj) --> [sky];

     to regions [3,4,6,7,8,9,10,14,15,16,17]

     SLOP fact retrieved: has_hue(green,[3,4,6,7,8,9,10,14,15,16,17]) :[0.009906
94,0.207021]

     SLOP fact determined: has_val(value@$low_val,[3,4,6,7,8,9,10,14,15,16,17])
:[0.842552,0.908525]

     SLOP fact determined: adj([3,4,6,7,8,9,10,14,15,16,17],[2,11,13]) :[1,1]


     SLOP fact determined: trees([3,4,6,7,8,9,10,14,15,16,17]) :[0.800675,1]

     Applying schema for trees

     to regions [5,1,12]

     SLOP fact retrieved: has_hue(green,[5,1,12]) :[5.19278e-05,0.0146256]

     SLOP fact determined: has_val(value@$low_val,[5,1,12]) :[0.98001,0.995545]

     SLOP fact determined: adj([5,1,12],[2,11,13]) :[1,1]

     SLOP fact determined: trees([5,1,12]) :[0.8352,1]

     Applying schema for road i.e.:

     schema(road,*x37) <~
             [road:*x37] -
                     (part) --> [vehicle]
                     (part) --> [tarmac]
                     (has_shape) --> [extended];

     to regions [18,19,20,21,22,23]

     SLOP fact retrieved: has_shape(extended,[18,19,20,21,22,23]) :[0.915675,0.981105]


     Subplan formed for area [18,19,20,21,22,23] is

     [(18,[]),(22,[20,21]),(23,[]),(19,[])]

     Applying schema for tarmac i.e.:

     schema(tarmac,*x41) <~
             [tarmac:*x41] -
                     (has_hue) --> [grey]
                     (has_val) --> [value@$low_val];

     to regions [18]

     SLOP fact determined: has_hue(grey,[18]) :[0,0]

     SLOP fact determined: has_val(value@$low_val,[18]) :[1,1]

     SLOP fact determined: tarmac([18]) :[0.4,1]

     Applying schema for tarmac

     to regions [22,20,21]

     SLOP fact determined: has_hue(grey,[22,20,21]) :[0.825301,0.959008]

     SLOP fact determined: has_val(value@$low_val,[22,20,21]) :[0.00408851,0.0783849]


     SLOP fact determined: tarmac([22,20,21]) :[0.495181,1]
```

```
Applying schema for tarmac

to regions [23]

SLOP fact determined: has_hue(grey,[23]) :[0.849246,0.979899]

SLOP fact determined: has_val(value@$low_val,[23]) :[0.715832,0.933694]

SLOP fact determined: tarmac([23]) :[0.509547,1]

Applying schema for tarmac

to regions [19]

SLOP fact determined: has_hue(grey,[19]) :[0.883045,0.987616]

SLOP fact determined: has_val(value@$low_val,[19]) :[0,0]

SLOP fact determined: tarmac([19]) :[0.529827,1]

SLOP fact determined: road([18,19,20,21,22,23]) :[0.899313,1]

Applying schema for trees

to regions [25,24]

SLOP fact retrieved: has_hue(green,[25,24]) :[0.105752,0.577313]

SLOP fact determined: has_val(value@$low_val,[25,24]) :[0.958712,0.991509]

SLOP fact determined: adj([25,24],[2,11,13]) :[0,0]

SLOP fact determined: trees([25,24]) :[0.830072,1]

possible car in regions with minX,maxX,minY,maxY as shown:-

region  minX  maxX  minY  maxY
22      29    63    58    66
20      22    49    52    61
21      37    51    57    59

What is support for?
|: 1

What is support against? (between 0 and 0)
|: 0
SLOP fact determined: road([18,19,20,21,22,23]) :[0.929984,1]

possible car in regions with minX,maxX,minY,maxY as shown:-

region  minX  maxX  minY  maxY
19      107   117   51    57

What is support for?
|: 0.9

What is support against? (between 0 and 0.1)
|: 0.1
SLOP fact determined: road([18,19,20,21,22,23]) :[0.995079,1]

yes
| ?-
```