

**EXAMINATIONS — 2007**

**END-YEAR**

**COMP 307**

**WITH SOLUTIONS**  
**INTRODUCTION TO**  
**ARTIFICIAL INTELLIGENCE**

\*\*\*\*\*

\*\*\*\*\*

**Time Allowed:** 3 Hours

**Instructions:** There are a total of 180 marks on this exam.  
Attempt all questions.  
Calculators may be used.  
Non-electronic foreign language translation dictionaries may be used.  
The appendices on pages 25–29 can be removed for reference.

**Questions**

1. Search [35]
2. Machine Learning General Concepts [17]
3. Nearest Neighbour, Naïve Bayes and Decision Trees [25]
4. Perceptrons and Neural Networks [20]
5. Evolutionary Computing and Learning [28]
6. Rule Based Systems [35]
7. Natural Language Processing [20]

## Question 1. Search

[35 marks]

(a) [4 marks] Search problems can be categorised into two groups: constructing problems and modifying problems. State two example search problems for each category.

constructing problems: 15 puzzle, route finding  
modifying problems: design a timetable, 8 queens

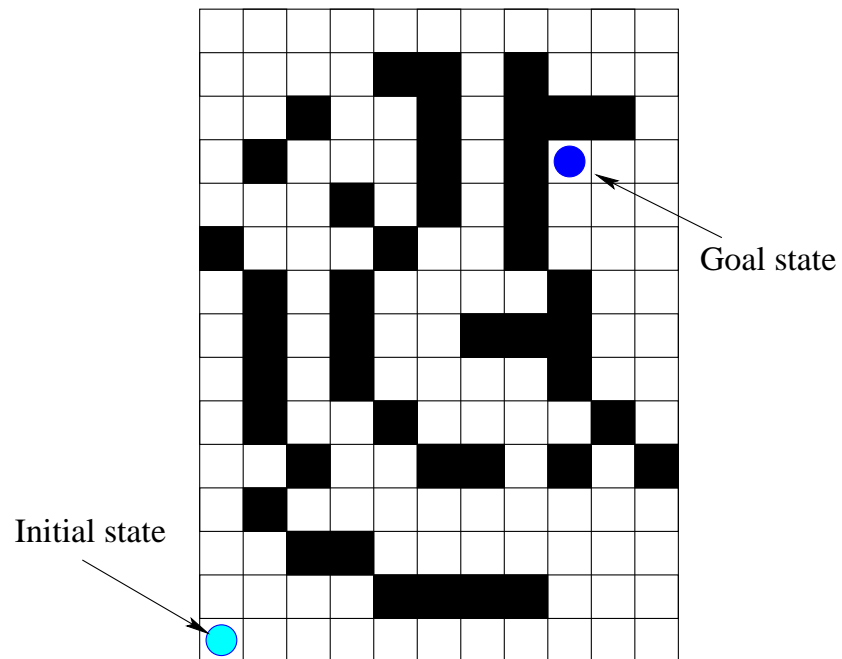
(b) [6 marks] Considering all the search algorithms introduced in the lectures, including *BFS*, *DFS*, *iterative deepening*, *PFS(priority first)*, *hill climbing*, *beam search*, *brunch and band*, and *A\**, identify a good algorithm that is suitable for constructing problems and a good algorithm that is suitable for modifying problems. Justify your answer.

constructing problems: A\* if heuristics can be found. Iterative deepening otherwise.  
Modifying problems: hill climbing if there is heuristics; otherwise, DFS.

(c) [6 marks] Briefly describe each of the following search algorithms:

- (i) Iterative deepening
- (ii) Priority first search

Consider the path finding problem in a maze for parts (d), (e) and (f). Suppose we want to find the shortest path from the initial state to the goal state.



(d) [6 marks] Outline an appropriate way of representing the search space — how would you represent the initial state, the goal state, and the actions that transfer one state to another?

X, Y coordinates  
Initial state: (0,0), goal state(8,11)  
actions: up, down, left, right

(e) [7 marks] Identify a good search algorithm for solving this problem and justify your answer. Explain your heuristics using examples.

A \*. good heuristics can be found. Path cost is the length of the path  
Heuristics: path length so far + city block distance to goal

(f) [6 marks] Draw a path on the figure on the facing page to show a solution found by your algorithm identified in the above question. Draw another path on the figure to show a path that might be found by DFS (using a different colour or dashed line). State any assumptions you made.

Draw on the figure!!!

## Question 2. Machine Learning General Concepts

[17 marks]

(a) [4 marks] Briefly describe the key difference between *supervised learning* and *unsupervised learning*, and state an example for each of them.

In supervised learning, the instances are labelled;  
in unsupervised learning, the instances are not labelled  
classification vs clustering

(b) [5 marks] State the name of an algorithm or approach used in each of the following machine learning paradigms:

- (i) Case based learning
- (ii) Induction learning
- (iii) Statistical learning
- (iv) Connectionist learning
- (v) Evolutionary learning

(i) Nearest Neighbour, (ii) decision trees, (iii) Naive Bayes, SVMs, (iv) neural networks, perceptrons, (v) GAs, GP...

(c) [4 marks] In machine learning systems, the data sets are typically separated into a *training set* and a *test set*. Briefly describe the role for each of them.

(i) A training set is a collection of instances that are directly used for learning a machine/concept/classifier,  
A test set is a collection of instances that are used to measure the performance of the learned machine/concept

(d) [4 marks] Briefly describe the *K-fold cross validation method* for classification in machine learning.

chop the available data into  $K$  equal chunks;  
For each chunk in turn, treat it as the test set, treat other  $K-1$  chunks as the training set, and the classifier trained from the training set is applied to the test set;  
The process is repeated  $K$  times with each chunk used exactly once as the test test;  
The  $K$  results are “averaged”.

**Question 3. Nearest Neighbour, Naïve Bayes and Decision Trees**

[25 marks]

(a) [3 marks] Why is  $R_i$  necessary in most cases in the weighted Euclidean distance measure that is often used in the *nearest neighbour method*:

$$distance = \sqrt{\sum_{i=1}^n \frac{(a_i - b_i)^2}{R_i^2}}$$

Note: the two instances A and B are represented by two feature vectors with numeric values:  $A = (a_1, a_2, \dots, a_n)$  and  $B = (b_1, b_2, \dots, b_n)$ ,  $n$  is the number of features, and  $R_i$  is the range of the  $i$ th feature.

| If such a range was not used, then the algorithm will be biased toward the features with very big values, which is often incorrect.

| In most situations, the relative importance of the features are unknown. It is reasonable to assume the features are equally important.

(b) [9 marks] Suppose you are building a Naïve Bayes spam filter to distinguish spam messages from real email messages. You have picked two key words: “medicine”, and “assignment” to characterise each message, and have counted how many of the messages contain each word:

	spam		email	
	word present	word not present	word present	word not present
“medicine”	50	350	1	49
“assignment”	1	399	20	30
Total count	400		50	

(Question 3 continued on next page)

**(Question 3 continued)**

If your spam filter was presented with a new message that contained both the words “medicine” and “assignment”, would your spam filter classify the message as spam or as email? Show your working.

$$\text{score(spam)} = 50/400 * 1/400 * 400/450 = 50/(400*450) = 1/3600$$

$$\text{score(email)} = 1/50 * 20/50 * 50/450 = 1/(5*225) = 1/1125$$

Therefore, it will choose email

(c) [3 marks] The decision tree learning algorithm uses an impurity measure to choose between attributes. Explain why the  $p(A)p(B)$  formula is a reasonable impurity measure for a set of instances belonging to two different classes  $A$  and  $B$ .

$p(A)p(B)$  is 0 if all the instances are in one category, and is a maximum if the two categories are equal, which is when they are equally mixed, and it is smooth in between.

(Question 3 continued on next page)

(Question 3 continued)

(d) [10 marks] Consider the following data set describing 10 loan applications at a bank, of which 5 were approved and 5 were rejected. They are described by three attributes: whether the applicants have a job or not; whether their deposits are low or high; and whether their credit records are very good, good or bad.

Instance	Job	Deposit	Credit	Class
1	true	low	very good	Approve
2	true	low	good	Approve
3	true	low	very good	Approve
4	true	high	very good	Approve
5	false	high	good	Approve
6	false	low	good	Reject
7	false	low	good	Reject
8	true	low	bad	Reject
9	true	low	good	Reject
20	false	high	bad	Reject

The bank wants to build a decision tree to help making grant loan decisions. Which attribute should the bank choose for the root of the decision tree if they use the impurity function  $p(\text{Approve})p(\text{Reject})$ . Show your working.

$$\text{Job: } 6/10 * (4/6 * 2/6) + 4/10 * (1/4 * 3/4) = 5/24 = 21\%$$

$$\text{Deposit: } 7/10 * (3/7 * 4/7) + 3/10 * (2/3 * 1/3) = 5/21 = 24\%$$

$$\text{Credit: } 3/10 * (3/3 * 0/3) + 5/10 * (2/5 * 3/5) + 2/10 * (0/2 * 2/2) = 6/50 = 12\%$$

Credit has the lowest score, therefore the algorithm will use

Credit at the root

Student ID: .....

**SPARE PAGE FOR EXTRA ANSWERS**

Cross out rough working that you do not want marked.  
Specify the question number for work that you do want marked.

#### Question 4. Perceptrons and Neural Networks

[20 marks]

(a) [4 marks] The XOR problem can be considered a binary classification problem where the output value is  $1$  (class1) when the two input variables have different Boolean values, and the output is  $0$  (class2) when the two input values are the same.

John Smith used a perceptron for the XOR problem. His perceptron had two input variables and a single output. However, his perceptron would not converge no matter how he changed the learning parameters.

Explain why John's perceptron was not successful.

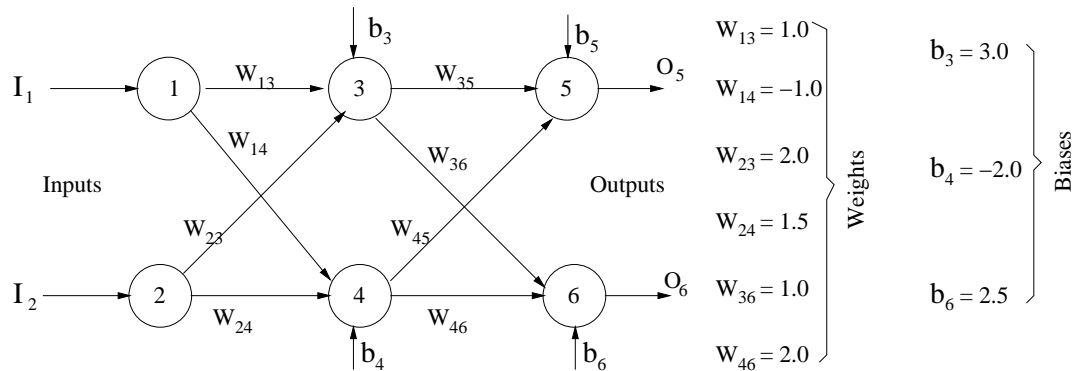
The instances are not linearly separable (by a hyperplane)  
the perceptron learning algorithm can only classify instances that are linear separable

(b) [6 marks] When using neural networks to solve classification problems, one common problem is *overfitting*.

- (i) Briefly explain the term *overfitting*.
- (ii) List three possible reasons that can cause this problem.
- (iii) State one approach that can avoid overfitting.

(i) The performance of the learned concept is very good on the training set, but is bad on the test set  
(ii) Training is too long, too few examples, too many hidden nodes/layers  
(iii) using a validation set

(c) [10 marks] Consider the following feed forward network that uses the sigmoid/logistic transfer function (see Appendix B),



- (i) What will be the output of node 6 ( $O_6$ ) for the input vector (0.0, 0.0)?
- (ii) What will be the new value of weight  $w_{36}$  after one epoch of training using the back propagation algorithm? Assume that the training set consists of only the vector (0.0, 0.0, 0.0, 0.0) and that the learning rate  $\eta$  is 0.5.

$O_1 = I_1 = 0; I_3 = b_3 = 3.0; O_3 = f(3.0) = 0.95; O_4 = f(-2.0) = 0.12$   
 $I_6 = 0.95 \cdot 1.0 + 0.12 \cdot 2.0 + 2.5 = 3.69; O_6 = f(3.69) = 0.97$   
 $\Delta W_{36} = \eta O_3 O_6 (1 - O_6) \beta_6 = 0.5 \times 0.95 \times 0.97 (1 - 0.97) (0 - 0.97) = -0.0134$   
 $(W_{36})_{new} = (W_{36})_{old} + \Delta W_{36} = 1 - 0.0134 = 0.9866$

### Question 5. Evolutionary Computing and Learning

[28 marks]

(a) [3 marks] Genetic algorithms and genetic programming are two techniques in evolutionary computing and learning. State three additional techniques in evolutionary computing and learning.

Evolutionary strategies, learning classifier systems, Ant colony optimisation  
Evolutionary programming, PSO, artificial life, artificial immune systems  
Differential evolution (DE), evolutionary multiobjective optimisation (EMO)

(b) [6 marks] In evolutionary learning algorithms, three genetic operators (*elitism*, *crossover* and *mutation*) are often used during the evolutionary process to generate new candidate solutions. Briefly describe each operator and its main purpose.

Elitist or reproduction, is the operator that directly copies a small set of best individuals from the current generation to the next generation. The main purpose is to make sure the evolution does not make the best solution worse.  
Crossover is the operator that combines genetic materials from different individuals. The goal is to form better individual solutions.  
Mutation is the operator that randomly changes a part of the selected individual. The main purpose is to maintain the diversity of a population.

(Question 5 continued on next page)

**(Question 5 continued)**

(c) [3 marks] Suppose that you want to use genetic programming (GP) for a particular task and that a GP package is available for you to use. Suggest three criteria for terminating a GP run.

<p>Set a maximum number of generations</p> <p>Early stopping: when the problem has been solved</p> <p>When the fitness of the best individual program cannot be improved over a certain number of generations</p>
---

(Question 5 continued on next page)

**(Question 5 continued)**

**(d)** [6 marks] Suppose you are using genetic programming to find programs to fit data from polynomial functions such as  $y = x^6 + 2x^4 + x^2 + 3.5$  given the value of  $x$ .

- (i)** Suggest a good terminal set.
- (ii)** Suggest a good function set.
- (iii)** Suggest a good fitness function.

Terminal set: { X, R }, R is a random number  
Function set: { +, -, \*, % } or { +, -, ^ }  
Fitness measure: TSS, MSE, ...

(Question 5 continued on next page)

**(Question 5 continued)**

(e) [10 marks] The standard tree based genetic programming approach has been applied to many classification tasks. In this approach, each evolved program typically returns a single floating point number, which is translated into class labels. For multiple class classification problems, one commonly used method for this translation is the *program classification map*, which splits the program output space into predefined regions, each corresponding to a particular class.

- (i) State two problems with this translation method.
- (ii) Suggest two distinct methods to overcome (or at least reduce) the problems and discuss each of the methods.

Limitations: the class boundaries are fixed  
the boundaries need to be predefined  
class orders are fixed

Two methods: (1) decompose the multiclass classification problem into multiple binary classification problem, then use GP each for each binary classification subproblem; (2) use dynamic class boundary determination methods

**Question 6. Rule based systems**

[35 marks]

Consider the following set of rules:

1. F if R and A and NOT N.
2. F if S and C and A.
3. P if S and C and H.
4. P if R and N.
5. S if L and NOT Y.
6. S if B and T.
7. R if Y and NOT T.

Suppose the following facts are true: B, C, H, L, N, A, T, Y

You may make your own assumptions as long as you include them in your answers.

(a) [8 marks] Suppose we use backward chaining and the hypothesis is tested in the order of F and then P. Explain how it reaches a conclusion.

rule 1, rule 7 fail, fail  
rules 2, rule 6, success  
So F is the conclusion

(b) [6 marks] Using the same set of rules and facts, explain how forward chaining will reach a conclusion.

rule 1, 2, 3, 4, 5, 6, success, add S to the memory.  
rule 1, 2, success, add f to memory.  
So F is the conclusion

(c) [6 marks] State two example tasks, one suitable for applying backward chaining and the other suitable for forward chaining. Briefly describe the reasons for your choices.

degree checking in A4, backward chaining, a lot of facts, most irrelevant, fewer conclusions  
 medical diagnosis, forward chaining, many facts known at the beginning, many conclusions

(d) [15 marks] In assignment 4, you wrote rules to check the BSc degree and some majors. Add rules so that it can check the BSc degree with a Math major.

Math major requirements:

- MATH 113, 114; one course from (COMP102, STAT 131/193, QUAN102); and
- at least 92 points from MATH 200-399 (excluding MATH 371, which is 12 points), including at least 48 points numbered 300-399.

The Utility Rules (Appendix C) and Documentation of the Rule Language (Appendix D) for assignment 4 are given at the end of the exam paper, which you may remove for reference if you wish.

```
hasMajor[?Student, math]
  IF allcourses[?student, ?courses]
  AND member[math113, ?courses]
  AND member[math114, ?courses]
  AND passedFrom[?courses, [comp102, stat131, stat193, quan102], 1]
  AND pointsFromDisciplines[?student, [math], 200, 399, 92]
  AND pointsFromDisciplines[?student, [math] 300, 399, 48]
  AND checkMath371[?student, ?courses]
  checkMath371[?student, ?courses]
  IF member[math371, ?courses]
  AND pointsFromDisplines[?student, [math], 200, 399, 104]
  checkMath371[?student, ?courses]
  IF NOT member[math371, ?courses]
```

## Question 7. Natural Language Processing

[20 marks]

Consider the following DCG grammar rules and some lexicons.

s --> np\_sub, vp, pp.

vp --> v, np\_obj.

vp --> v.

pp --> prep, np\_obj.

np\_obj -> pro\_obj.

np\_obj > det, n.

np\_sub -> pro\_sub.

np\_sub > det, n.

pro\_sub --> [i].

pro\_sub --> [you].

pro\_sub --> [he].

pro\_sub --> [she].

pro\_sub --> [they].

pro\_sub --> [john]

pro\_sub --> [mary].

pro\_obj --> [me].

pro\_obj --> [you].

pro\_obj --> [him].

pro\_obj --> [her].

pro\_obj --> [them].

pro\_obj --> [john].

pro\_obj --> [mary].

prep --> [to].

prep --> [by].

det --> [the].

det --> [a].

n --> [cake].

n --> [butter].

v --> [loves].

v --> [saw].

v --> [gave].

v --> [ate].

- (a) [6 marks] Show the parse tree for the following sentence:  
“Mary gave the cake to him”

np-sub (pro-sub (Mary), vp ( v(gave), np-obj ( det(the), n(cake)) , pp (prep(to),  
np-obj(pro-obj(him)))

- (b) [6 marks] Write six sentences that are considered correct using the grammar. Construct them so that two are syntactically correct, two are syntactically incorrect and two are semantically incorrect (something do not happen in the real world). Label the sentences clearly using “correct”, “incorrect”, “semantically incorrect”.

correct: John loves mary, she saw him  
incorrect: I loves mary, He gave a butter to her  
semantically incorrect: he ate mary, the cake ate mary

**SPARE PAGE FOR EXTRA ANSWERS**

Cross out rough working that you do not want marked.  
Specify the question number for work that you do want marked.

(c) [8 marks] What are the main problems of this simple grammar? Briefly explain how they can be fixed.

it does not check the subject, verb agreement  
it does not check the det noun constraints  
Add the grammatical features to the lexicon  
Rewrite rules by augumetn the rules with variables, to check the constraints.

\*\*\*\*\*

**SPARE PAGE FOR EXTRA ANSWERS**

Cross out rough working that you do not want marked.  
Specify the question number for work that you do want marked.

## Appendix for COMP307 exam

(You may tear off this page if you wish.)

### A Some Formulae You Might Find Useful

$$p(C|D) = \frac{p(D|C)p(C)}{p(D)} \quad (1)$$

$$f(x_i) = \frac{1}{1 + e^{-x_i}} \quad (2)$$

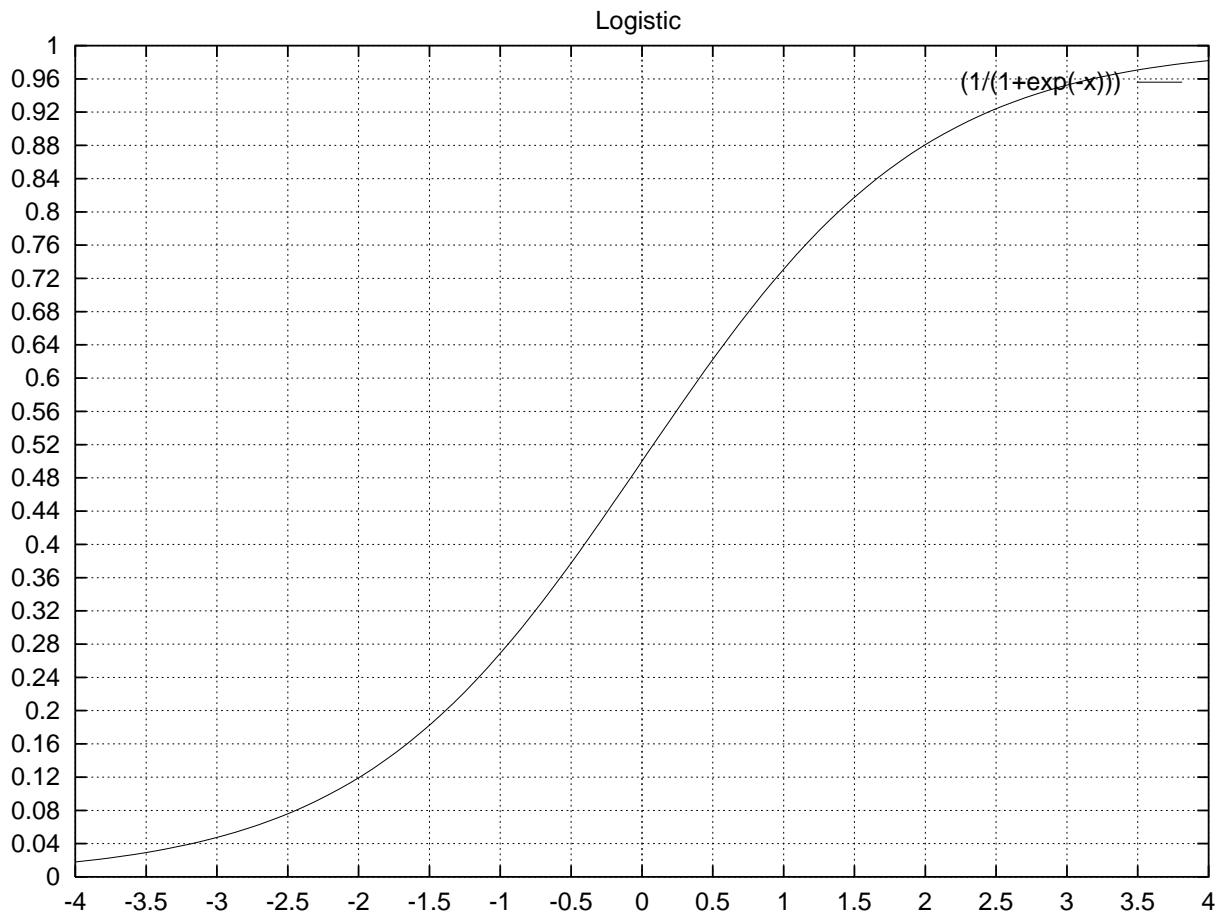
$$O_i = f(I_i) = f\left(\sum_k w_{k \rightarrow i} \cdot o_k + b_i\right) \quad (3)$$

$$\Delta w_{i \rightarrow j} = \eta o_i o_j (1 - o_j) \beta_j \quad (4)$$

$$\beta_j = \sum_k w_{j \rightarrow k} o_k (1 - o_k) \beta_k \quad (5)$$

$$\beta_z = d_z - o_z \quad (6)$$

## B Sigmoid/Logistic Function



**C Utility Rules for checking degree and majors**

```

# Utility Rules -----

# What are all the courses a student has passed
allcourses[?student, ?courses]
  IF bind[?courses, executeSqlList["SELECT coursecode
    FROM Students
    WHERE id='",?student,"'"]].

# Are there at least ?num of the courses in ?lst in the list of all passed ?courses
passedFrom[?courses, ?lst, ?num]
  IF greaterOrEqual[size[intersection[?courses, ?lst]], ?num].

# Has the student passed at least n points in a range of course numbers
# (eg 200 to 399)
pointsFromRange[?student, ?min, ?max, ?required]
  IF bind[?points, executeSqlSingle["SELECT SUM(points)
    FROM Students NATURAL JOIN Courses
    WHERE id='",?student,"' AND number>=",?min, "
    AND number<=", ?max]]
    AND greaterOrEqual[?points, ?required].

# Has the student passed at least n points from courses in one of a given list
# of disciplines and range of course numbers
pointsFromDisciplines[?student, ?disciplines, ?min, ?max, ?required]
  IF bind[?points, executeSqlSingle["SELECT SUM(points)
    FROM Students NATURAL JOIN Courses
    WHERE id='", ?student,"'
    AND discipline IN (" , ?disciplines, ")
    AND number>=",?min, "
    AND number<=",?max]]
    AND greaterOrEqual[?points, ?required].

# Has the student passed at least n points from courses in a given degree schedule
#and range of course numbers
pointsFromSched[?student, ?schedule, ?min, ?max, ?required]
  IF bind[?points, executeSqlSingle["SELECT SUM(points)
    FROM Students NATURAL JOIN Courses
    WHERE id='",?student,"'
    AND schedule='",?schedule,"'
    AND number>=",?min, "
    AND number<=",?max]]
    AND greaterOrEqual[?points, ?required].

```

## D Documentation: Rule Language

The language for the queries, facts, and rules is a simple first-order language with variables. The facts are predicate terms that assert that some relationship holds between some values. A fact consists of a predicate name followed by the arguments in square brackets. The arguments may be constants (numbers or identifiers) or lists (which are enclosed in square brackets). For example, the following are predicate term facts

```
isa[object1, block].
length[brick1, 15].
contains[arch1, [brick1, brick2, brick3]].
```

Like facts, the queries are predicate terms but may also contain variables (identifiers starting with a "?"). If a query contains a variable, the backward-chainer will attempt to find a binding for the variable that makes the query true (according to the facts and rules). For example:

```
isa[?obj, block]
isa[brick1, ?category]
length[brick2, ?length]
length[?obj, 21]
hasDegree[301023, bsc, comp]
failsCode[woods, ?reason]
```

Rules consist of a head and a set of conditions. Like a query, the head is a predicate term that may contain variables. The conditions of a rule are a sequence of literals, connected by "and". Each literal is a predicate term, a Boolean operator term, or the negation of a term. The arguments of condition terms may also contain non-boolean operator terms (listed below). For example, the following are two rules that describe when an object consisting of three bricks forms an arch:

```
isa[?object, arch]
  IF contains[?object, [?br1, ?br2, ?br3]]
  AND on[?br3, ?br1]
  AND on[?br3, ?br2]
  AND archProportions[?br1, ?br2, ?br3]
  AND NOT touching[?br1, ?br2].

archProportions[?sup1, ?sup2, ?top]
  IF length[?sup1, ?h1]
  AND length[?sup2, ?h2]
  AND equal[?h1, ?h2]
  AND width[?sup1, ?w1]
  AND width[?sup2, ?w2]
  AND length[?top, ?lt]
  AND greaterThan[?lt, sum[?w1, ?w2]].
```

When the user enters a query, the rule system will attempt to match the query with a fact or the head of a rule from its rule set (in order of their occurrence in the fact/rule set). If the query matches a fact, then it will return success, along with any variable bindings required to make the match. If the query matches the head of a rule, then it will attempt to satisfy all the conditions in the rule (in order), accumulating bindings of variables as it goes. When a rule condition is a predicate term, it attempts to satisfy the term in exactly the same way as the original query. When a rule condition is a boolean operator term, it will attempt to evaluate the operator, and evaluate any of the arguments of the operator along the way. Except for the bind operator, operators will evaluate all their arguments. This means that any variables in an operator term must have been bound to a

value by the time the operator term is considered. For example, in the `archProportions` rule above, `?lt`, `?w1`, and `?w3` must have been bound by the last condition (which they will have been). Note that any operators inside a predicate term will not be evaluated when the predicate term is examined.

The boolean operators are:

- `greaterThan[value, value]` and `greaterOrEqual[value, value]`: binary operators that compare numbers.
- `notEqual[value, value]`: binary operator that compares any values
- `member[value, listvalue]`: binary boolean operator that returns true if the first argument is an element of the second (list-valued) argument.
- `bind[variable, value]`: binary boolean operator that fails if the value is fail or if the variable is already bound to something different from the value. Otherwise, it succeeds (returns true) and also binds the variable to the value.
- `ask[predicate term]`: asks the user whether the term is true or false (unless the user has already been asked the same question, in which case it returns their previous answer without bothering them again).

The non-boolean operators (which return a value) are:

- `sum`, `difference`, `product`, and `quotient`: binary numeric-valued operators that return the sum, difference, product and the quotient of their numerical arguments.
- `intersection`, `union`: binary list-valued operators that return the intersection or union of two lists.
- `size`: unary numeric operator that returns the length of a list.
- `executeSqlSingle[SQL query]`: Invokes the SQL query on the `comp307` postgres database and returns the first value of the list of values that the SQL query returned. The SQL query must return just a single field or number. If the SQL query returns no values, the operator returns the value fail. The arguments of this operator (and the next), after they have been evaluated, will be concatenated into a single string that will be the SQL query. Typically, the arguments will be a mixture of strings (with most of the SQL query) and variables, which must be bound to values that will be inserted into the SQL query
- `executeSqlList[SQL query as a string]`: Invokes the SQL query on the `comp307` postgres database and returns the first value of the list of values that the SQL query returned. The SQL query must return just a single field or number. If the SQL query returns no values, the operator returns the value fail.